Title: Chinese Character Description Languages (CDL)

Source: Richard Cook <rscook@unicode.org>

Status: Expert Contribution
Date: 2003-10-26-22:22
Action: For consideration by UTC and IRG

The current document is an extract of nine pages from my 2003 PhD Dissertation. These pages are intended to introduce some aspects of Wenlin Institute's CDL, a system which holds much promise for management of CJK Unified Ideographic character data. This document augments part three of L2/03-286 (Cook & Bishop). Additional documents in this series will follow, including a draft of the full XML CDL specification.

References

Cook, Richard S. 說文解字-電子版 *Shuo Wen Jie Zi - Dianzi Ban: Digital Recension of the Eastern Han Chinese Grammaticon*. UC Berkeley, Dept. of Linguistics, 2003.
L2/03-286 <http://www.unicode.org/L2/L2003/03286-cook2.txt>.
L2/02-221 <http://www.unicode.org/L2/L2002/02221-cdp-idc.pdf>.

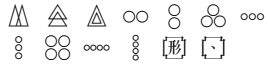
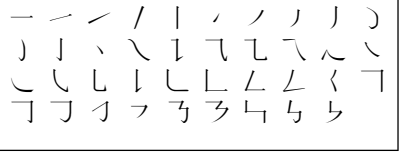
Abbreviations

SW = *Shuo Wen Jie Zi* (121 AD text, see above).
HDZ = *Hanyu Da Zidian* (see the kHanYu header in Unihan.txt).
SBGY = *Song Ben Guang Yun* (see the kSBGY header in Unihan.txt).

2.5.5 Chinese Character Description Languages (CDL)

In order to quantify the relations among character forms, to enable ancient texts to "talk to each other", I have employed three Chinese Character Description Languages (CDL)[41] in association with SW and Unicode-based variant mapping tables. One of these CDL's, the CDP system developed at Academia Sinica in Taiwan is a Big5-based component system. The second is the IDS system of the Unicode Standard. The third, and by far the most advanced system that I am aware of, is a stroke and component-based system being developed by Wenlin Software (Bishop, 2003), to which I am (and have been) contributing, with special regard to extending its applications for computer standards work. Some basic information about these three CDL's is tabulated in Table 2-27, along with sample elements used in each.

Table 2-27. Some Elements of Three Chinese Character Description Languages

Name	*Creator*	*Elements*
CDP	*Chinese Document Processing Lab., Institute of Information Technology, Academia Sinica, Taiwan*	
IDS	*Unicode 4.0, Unicode Consortium*	
CDL	*Wenlin 3.x, Wenlin Institute*	

The CDP and IDS systems both have extreme limitations, both in regard to their intended purposes and also in regard to their basic elements. For this reason, they are not discussed here, and the reader is referred to the Glossary entries for CDP and IDS for further details. Due to legacy data issues, the CDP system was however employed by me in preparation of the HDZ and SBGY data appearing in the present study, and for this reason this system is discussed in some detail in Section 3.2.3.2.

41. The name "CDL" for WL's stroke-based system was coined in a discussion I had with Kenneth Whistler in 2002.

Wenlin's stroke-based system has the potential to become a full-fledged CDL such as might be adequate for handling all encoding issues, and though we cannot look at details of its software implementation here, some of the distinctions which it makes serve as the basis for the full CDL described here.

2.5.5.1 An Extensible Set of Basic Script Elements for *Han*

An extensible set of basic script components for *Han* is envisaged as a means for quantifying the relationships among characters and also among glyph variants, for indexing and encoding purposes, and for the purpose of building variant tables to be used for investigating the inter-relations among texts and inscriptions.

The set of basic stroke types listed above in Table 2-10 (repeated in Table 2-27), augmented with other more rare basic stroke types constitutes the basic set of distinctive features. The members of this set, used in accordance with a standard Cartesian coordinate grid (rather than the CDP or IDS type of spatial relation operators), and in association with a few transformations necessary for rare characters (*cf.* Table 2-11), provide a means for unambiguous mathematical description of all Chinese characters. By means of such descriptions, it is possible to automate the identification of component structures, and to quantify the differences among character forms.

These basic script elements and their associated transformations (treating positioning, scaling, flipping and other stroke modifications all as "transformations") altogether constitute the set elements, and this set is "extensible" insofar as the only limits on set membership are practical ones. That is to say that if someone cares to make a distinction which has not already been made, then the CDL is able to accommodate addition of that new distinction. The addition of a rare stroke type would be one example of the CDL's extensibility. The addition of a "flip horizontal" transformation to the CDL (*cf.* Table 2-11) would be another example of its extensibility, in that the CDL does not at present have any such transformation.

2.5.5.2 CDL Descriptions: Examples

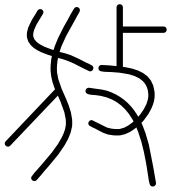
For particular examples of CDL descriptions, let's revisit in Table 2-28 below the forms first exhibited in Table 2-19:

Table 2-28. Members of the 敖 *áo* Graphical Variant Class (reprise)

			
{eci} == {gjb}	[U+22f8d]	[U+e109]	[U+6556]

The traditional componential analysis of the seal form gives us only two components for this character, as follows:

Table 2-29. Seal components of 敖 *áo*

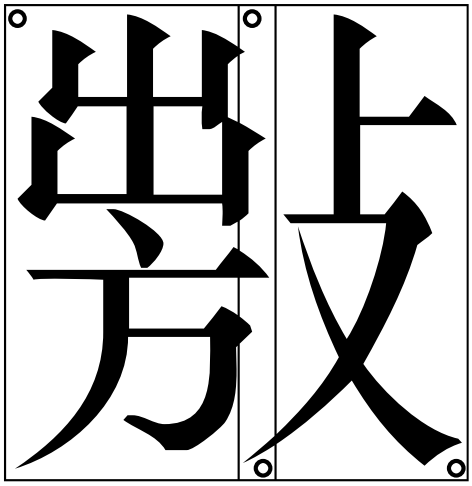
It is apparent here that the form of the component  {ech} /pʲɑŋʔ/ (313.04), /pʲɑŋs/ (426.48) in the compound graph has undergone 省 *sheng* 'contextual distortion' in combination with the {gja} /tɕʰiueis/ (356.05), /tɕʰiuet/ (474.46) component. This distortion exemplifies the distinction between *etymographical componential analyses* on the one hand, and *simple structural componential analyses* on the other. A *kai* representation of the above analysis in Table 2-29 would use simply 出 and 放 components.

And yet, attempting to give a structural representation of the form using a CDL of some type requires that the two parts of *one* of the etymographic parts of the character be transformed (scaled, distorted) *separately*. That is, in order to place 出 over 方 , we must first separate 方 out of the compound etymographic component 放 (which, by the way apparently has no encoded ultra-*fanti* form 放 with the full 攴 {dge} /pʰuk/ (452.18), /pʰok/ (466.09) component). This kind of transformation (independent scaling of separate parts of

a single component), while not programmatically impossible, is however rather inconvenient. Ideally, one might like to preserve the etymographic component analyses as much as possible in one's CDL descriptions, and yet it is rather more convenient to simply treat the two things distinctly, as separate though related tiers. SW's 省 *sheng* etymographic components often omit component elements entirely (rather than simply distort them). And as we have seen (Section 2.5.3), there is often no consensus on component analysis itself.

One solution is to simply ignore *theoretical* (etymographic) explanations of the character for the purposes of the CDL (etymographic information can be stored elsewhere), and worry only about the character's *actual* appearance in the specific context of its occurrence. This presents us with yet another problem, since there is in fact no such character as 㓁 (at least there may not have been until now).[42] Figure 2-1 below[43] illustrates the simple CDL description of two components in left-to-right combination. Note that the two components 㓁 and 攴 each have bounding boxes, and that each bounding box has "control dots" at its upper left and lower right corners (to control component scaling within the grid space).

Figure 2-1. 斆 CDL for Ultra-*fanti* 斆 [U+22f8d], with PUA component



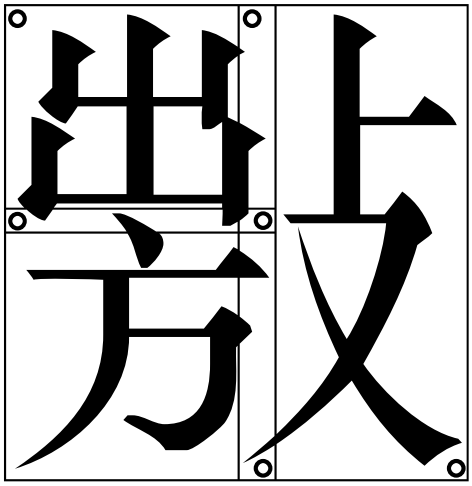
This structure 㓁 with 出 over 方 is not an independent element in the script, but might be identified as simply a highly bound graphical component (according to the usual left-to-

42. But cp. the left side of 𠂤 {gyy} /ʔɟɑnʔ/ (280.16), /ʔɟanʔ/ (295.19).

43. CDL Figures in this section are based on WL's implementation, and were produced using *Wenlin 3.x* and *Fontographer* 4.1.4 software; see *Bishop*, and *Altsys*, both in the Bibliography, and *fn. 45* below.

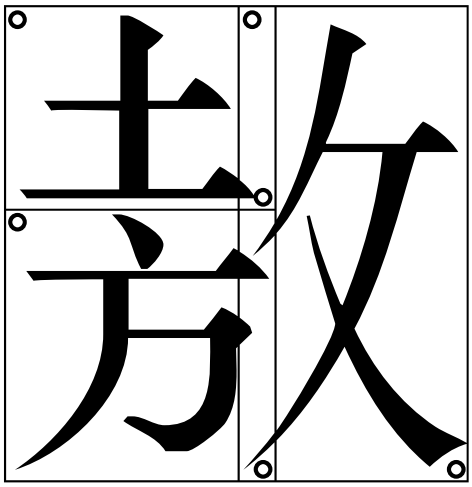
right structure of characters with the 攴 / 攵 radical). We can assign this 𣪊 form to PUA, as I have done for the present discussion: 𣪊 [U+e0c0]. And yet this type of PUA usage is rather pointless for highly bound structures which might be decomposed into non-PUA (in this case BMP) components, as in Figure 2-2.[44]

Figure 2-2. 斃 CDL for Ultra-*fanti* 斃 [U+22f8d], with BMP components



Similarly, for the second square-script form 𣪘 in Table 2-28, rather than defining a nonce PUA component for the left-hand side, we simply resort to elements of a somewhat lower-level description of this bound form of the BMP graph 敖 [U+6556]. See Figure 2-3.

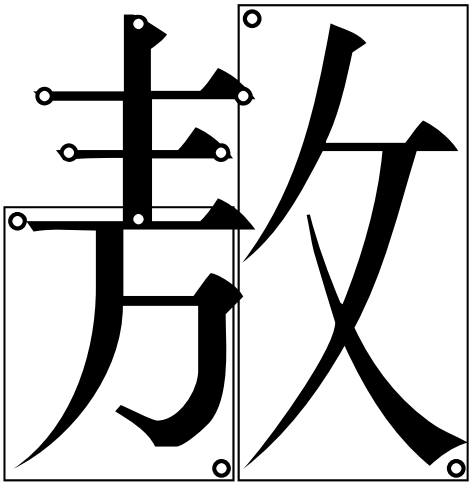
Figure 2-3. 敖 CDL for PUA graph 𣪘 [U+e109], with BMP components



44. To transform multiple components, a grouping mechanism might be used to associate components.

The CDL description of 敖 [U+e109] (PUA graphical variant of 敖 [U+6556]) given in Figure 2-3 above) specifies the components 土 [U+571f], 方 [U+65b9], and 攵 [U+6535], each with the (x, y) coordinates of its bounding rectangle (each with two “control dots” at the corners) within the grid space (bounding box) of the composite character as a whole. This description is not self-contained. Rather, in order to display the composite character, the language interpreter uses the CDL descriptions of each of the three components. In general, components can be any characters that are themselves defined as sequences of basic strokes and/or simpler components.[45]

Figure 2-4. 敖 CDL for BMP graph 敖 [U+6556], with 3 stroke components



Similarly, the CDL description for the Unicode 3.0/4.0 reference glyph of BMP graph 敖 [U+6556] specifies the first three strokes as basic stroke types 一, 一, and 丨, each with the coordinates of its starting and ending points (note the positions of the control dots at stroke extremities), and possibly using stroke modifiers (note stroke 3), and then specifies the two components 万 [U+4e07] and 攵 [U+6535] with their bounding rectangles.

All of the above CDL descriptions can of course be reduced all the way down to the stroke level, since all components are comprised of only basic stroke types to which trans-

45. This is an expansion of Cook 2003b; portions of the discussion of Figure 2-3 and Figure 2-4 derive from discussion with Thomas Bishop (see the acknowledgements in Cook 2003b).

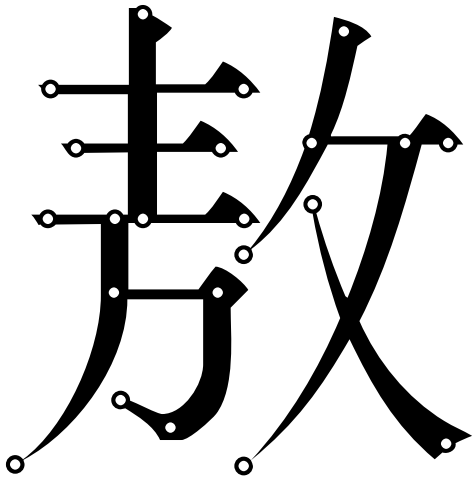
formations of various kinds may be applied. Thus, the 10 strokes of the graph for 敖 [U+6556] in Figure 2-4 can be reduced to the sequence given in Figure 2-5 below.

Figure 2-5. Sequence of strokes for Figure 2-4

一	一	丨	一	乛	丿	丿	一	丿	乀
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*

Figure 2-6 below presents the graph at its lowest level of decomposition, where all strokes are represented as independent entities (note the control dots at all stroke extremities, and some stroke types have more than two controls).

Figure 2-6. 敖 CDL for BMP graph 敖 [U+6556], with 10 stroke components



2.5.5.3 CDL Constraints

Bear in mind that the CDL for a given script entity is not simply a succession of loosely defined basic stroke types, but that each stroke type is rigorously defined as a specific sequence of straight and curved segments, that each stroke type has a specific range of behaviors (allowable transformations), and that in a particular usage instance each stroke type has associated features which define its context (coordinates and transformations).

Specific instantiations of a given type therefore provide sufficient information both for identification of the type and also for quantification of the degree of variation among members of a particular type class. So, for example, stroke type A is readily distinguishable from

stroke type B, due to the properties of each. And stroke type A with transformation type X is readily distinguishable from stroke type A with transformation type Y, due to the properties of the transformation.

Likewise, when a given type is employed in a particular composite structure, the presence or absence of the type, or the presence or absence of a particular transformation of the type, may be sufficient grounds for distinguishing the two composite structures.

Comparison of Figure 2-2 with Figure 2-3 may reveal, for example, that the differences between the CDL for 敷 and 款 lie in the upper left component, and in the form of the right component, and that otherwise all relative proportions (component transformations) are identical, as is the third component 方 . This information alone might be sufficient to indicate a possible relation among these two forms. If however additional information is added to the mix, such as information on allowable or known variant component shapes, then the possibility of connection becomes even stronger.

We know in this case that 攴 / 夂 component variation alone is never sufficient grounds for exclusion of a variant relation among two forms. If a computer program evaluating the possible difference among 敷 and 款 is given this additional information, then the suggested relation between the two forms would be quite strong.

2.5.5.4 CDL-Driven Inferences, VarClass Determinations, and Unifications

We might infer from this that 出 / 土 component variation is also evident among other encoded forms, *i.e.* that 土 is a simplification of 出 in other compounds as well. Searching our component data for characters with this 出 component, and looking at their variants in our variant mapping tables, we find that this is in fact the case. For example, in writings of characters with the 𡍮 崇 {adi} /siueis/ (351.01) component, we sometimes find this written as 𡈼 instead, as in writings of 𡎚 {izc} /kʰuɑnʔ// (285.48). Note the second and third members of the triple variants listed here: 歁 欯 款 .

From our initial inference regarding 出 / 土 variation, we now know that this can be extended to 出 / 土 / 木 component variation (note that variant mappings here and elsewhere are dependent upon the HDZ entries, in this case those for 歀 欵 款). We learn from this also that not only does 耒 sometimes vary with 奈 , but 奈 varies with 奈 in some compounds, so that the chain of related component forms has now become 耑 耒 奈 奈 . As independent characters, the preceding four forms may all have distinct usages, but in compounds the usage of one of these four in one text may be interchangeable with the usage of another of these in another text.[46]

It is clear then that variant mapping has implications not simply at the character-to-character level of mapping, but also at the *character-component*-to-*character-component* level of mapping. Variant unification (that is, the determination that there is non-distinctive variation among varclass members) can be undertaken at the component level, with either higher or lower level component descriptions.

Also, although the stroke order for Chinese characters is usually quite well defined, there are exceptional cases in which there are competing stroke orders. The CDL descriptions themselves are sensitive to stroke order, and yet stroke order might also be ignored for certain purposes, *e.g.* in variant mapping.

Up to this point we have seen elements of a CDL and how these elements may be employed to categorize the relations among script entities. We have also seen how such categorizations might be useful for certain purposes, including indexing forms, identifying, cataloguing and analyzing character variants. As we conclude this Chapter and move into the next, we shall consider specific extended examples of CDL usage for the purposes of historical linguistics.

46. To anyone who has worked on comparative semantic data (*e.g.* Tibeto-Burman gloss data in the STEDT databases), this kind of progression must seem familiar.

IDEOSY

An Ideographic and Interactive Program Description System[†]

Alessandro Giacalone, Martin C. Rinard, and Thomas W. Doeppner Jr.

Department of Computer Science
Brown University
Providence, Rhode Island 02912

1. Introduction

IDEOSY is an experiment in the use of a formal semantics as the basis for a programming system and in the use of an *ideographic* language as the primary means of user-computer communication. The important characteristics of our system are that it uses an ideographic syntax, has a syntax-directed editor, supports the definition of various equivalence properties and the proofs of such equivalence, and has an interpreter. It currently runs on Apollo workstations and on VAXes running Berkeley UNIX[‡] using any of a variety of high-resolution color displays.

Our formalism is based on Milner's Calculus of Communicating Systems (*CCS*) [1]. We have found CCS to be a convenient formalism for describing programs and have even used it for describing the UNIX operating system [2]. Its algebraic properties are very useful for building descriptions out of components and for proving the equivalence of descriptions. Since CCS is an operational semantics, we may directly interpret descriptions written in CCS.

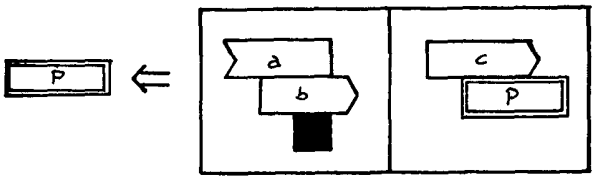
The idea behind using an ideographic interface such as ours is that a graphically suggestive language will aid the process of translating one's intuitive idea of a program's structure into a formal description. Our language, IDCCS (for *Ideographic Calculus of Communicating Systems*), was described in a previous paper [3]. It uses ideographs (pictures) to represent the various elements and operators of CCS. This orientation, i.e. the use of ideographs instead of words, is the cornerstone of our system. It allows rapid interaction between the user and the system, as the user may quickly manipulate descriptions by selecting ideographs representing its components. This ability to work with a formal description graphically is exploited not only in the editor but in the proof-aider and interpreter as well.

In the next few sections we first introduce the editor and the language. We discuss the concept of equivalence in CCS and then cover the support for formal reasoning in IDEOSY. This is followed by a discussion about the use of our interpreter. Finally, we discuss current limitations, extensions and future developments.

2. The Editor

We describe the editor of IDEOSY with the help of an example, trying as much as possible to show what the actual screen looks like. The editor can be used in two basic ways: to construct an expression in a top-down fashion, i.e. starting from scratch, and to modify an existing expression.

We illustrate the top-down part first by showing how we insert the definition of the agent named P below into the system and, in particular, how the defining behavior expression is constructed. The expression defines the behavior of P as that of an agent which can either execute a sequence of two communication actions (named *a* and *b*) and then terminate or execute an action named *c* and then reproduce its behavior.



The symbol

stands for an agent identifier. Juxtaposed boxes define, in the syntax of IDCCS, an "exclusive or" of the alternatives enclosed in each box. The symbols

[†]This work was partially supported by the National Science Foundation under grant MCS 8121806 and by the Office of Naval Research and the Defense Advanced Research Projects Agency under contract N00014-83-K-0146 and ARPA order No. 4786.

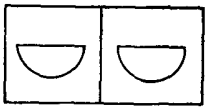
[‡]UNIX is a trademark of Bell Laboratories.

are ideographs for input and output actions, respectively. Actions arranged diagonally from upper left to lower right define a sequential execution of those actions. Finally, the symbol:

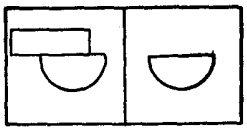
is an ideograph for a "null agent", i.e. an agent that does not perform any action, but represents the termination of an agent.

To define our expression, we inform the editor that we wish to define a process and provide a name for it (in this case P). At this point, the editor displays the symbol below:

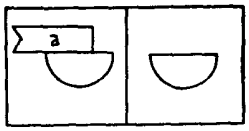
The "bowl" symbol is the *start* symbol in the IDCCS grammar as well as the non-terminal symbol (and ideograph) for a process. An expression is constructed by repeatedly replacing a non-terminal symbol with the right-hand side of a production of the IDCCS grammar. We specify which production we wish to apply by selecting ("picking") it from a menu of ideographs, which varies according to the type of non-terminal selected for replacement. Assuming that the "exclusive or" ideograph is selected, the picture below is displayed.



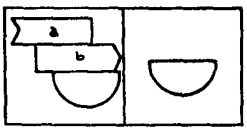
We must now "expand" the two bowl symbols in each alternative of the "or". Assume we expand the one on the left first. Since we want to produce a sequential expression, we select the corresponding ideograph and obtain the picture:



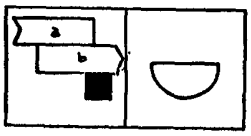
On the left we now have a construction formed by the ideograph for an *action* followed by the bowl. Assuming that we expand the action symbol first, we have to choose among the types of action (input or output). By choosing an input action and naming it a, we obtain the picture:



By repeating the procedure above for the bowl symbol appended to the input action just created, we can create an output action named b and obtain the picture:



At this point, we replace the bowl symbol with the symbol for a null process and obtain:



By proceeding analogously with the right side of the "or" we obtain the expression we wanted.

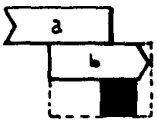
Besides the top-down expansion of non-terminals, it is possible to replace any construct that appears in an expression (i.e. any terminal, non-terminal, sub-expression, or the whole expression) with a construct of the same syntactic type. Summarizing, to replace one construct with another, one

(1) selects the construct to be replaced from the expression on the screen (possibly the whole expression);

(2) picks the *replace* command (it is a button always on the screen);

(3) identifies the new construct (the system will produce an error message if the new construct is not of the correct type).

The construct to be replaced (step (1)) is selected by picking any point within the smallest rectangle that encloses it (an operation called *range picking*). For example, in the expression below:



the dashed rectangle defines the area within which we must pick in order to select the sub-expression:

Terminal symbols (which are enclosed in rectangles) are a particular case and define a range on their own (so that in the picture above we would have to pick within the rectangle but outside the terminals, otherwise one of the terminals would be selected).

The replacing construct is selected (step (3)) by either picking it from an expression on the screen (by range picking) or selecting a definition among those

known by the system.

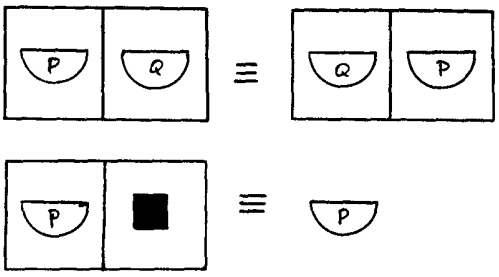
The rules that define which construct is range-picked in each case are non-ambiguous but not always immediate. However, the flexibility introduced by *replace* surpasses its drawbacks: for example, the "deletion" of a construct from an expression is a particular case of replacement and is actually implemented as a "derived operator" of *replace* (note that the system will not allow the deletion of a construct from an expression if the resulting expression is structurally incorrect).

3. CCS Equivalence Relations

In CCS, a number of equivalence relations are defined on behavior expressions: *identity* ($=$), *direct equivalence* ($\equiv$), *strong congruence* ($\sim$), *observation equivalence* ($\approx$) and *failure equivalence* ($\simeq$). The equivalence relations differ in how refined a partition they determine on the set of behavior expressions: stronger relations determine more refined partitions and smaller equivalence classes, thus imposing stronger conditions for two behavior expressions to be considered equivalent. The relations were listed above in decreasing order of "strength", in the sense that each relation *implies* all the following ones:

$$= \subset \equiv \subset \sim \subset \approx \subset \simeq$$

Some equivalence relations are *congruences*, i.e. are such that when two expressions are equivalent (with respect to that equivalence) then each expression can be replaced by the other in any operator context (typically, as a part of a more complex expression) without altering the meaning of the the entire expression. This is not true of all the equivalence relations defined in CCS. Those which *are* congruences are particularly useful. They allow the definition of "equations" (more properly, *equational laws*) between classes of processes (represented by behavior expressions). Examples of (simple) equational laws, expressed in IDCCS syntax, are:



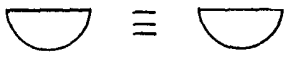
which define, respectively, the commutativity of the "or" operator and the possibility of eliminating null operands from an "or".

The equations can be used to apply semantics-preserving syntactic transformations to behavior expressions, for example to simplify an expression, or, more generally, to put an expression in a form which is "useful" for further reasoning. In the following, we refer to such equations simply as "rules". In IDCCS syntax rules are pairs of expressions which (in general) include non-terminal symbols of the grammar (possibly named) and therefore identify "classes" (not to be confused with *equivalence* classes) of expressions.

4. Support for Formal Reasoning

Rules such as those listed above constitute the basic knowledge IDEOSY has about CCS semantics: the system represents the equivalence relations defined on CCS behavior expressions as a list of rules appropriate for each equivalence relation. The system can apply the rules it knows and thus perform all the repetitive, and usually quite complicated, work involved in reasoning about CCS definitions. The user is then free to concentrate on more "strategic" decisions. Moreover, the system can support the user's decisions by detecting which of the rules it knows are relevant, i.e. could be applied in each case. A few basic rules are "hardwired" in IDEOSY. Any other rule which is found to be generally useful can be inserted in the system at any time.

A rule is inserted by using the editor of the previous section. For rules, the syntax that guides the editor is that of an equation, i.e. two expressions related by some equivalence symbol. For example, if one wanted to insert one of the rules shown above, the first picture displayed after the user picked the desired type of equivalence would be:



With this, the "editor" is telling the user (1) which type of equivalence the property being defined refers to, and (2) that the user has to define the two expressions.

When the editor is used to edit a rule, the user is allowed to name non-terminals. For example, in the two examples of rules shown above the commutativity of the "or" operator and the elimination of null operands from or's were defined by appropriately naming by P and Q the non-terminal symbols for processes (the bowls). The system interprets non-terminals in rules as "syntactical variables", i.e. as representatives of all the expressions they can generate via the applications of the productions of the grammar.

Once a rule is acquired, it is associated by the system with all of the terminal symbols that appear in it. Thus in the example above, the rule is associated with the symbol:

(being a property of the "or" operator), while in the case of the "*NIL*-elimination" rule mentioned above, the rule is associated with both the symbols

When the user wants to transform an expression, he or she can obtain a list of the rules applicable to sub-expressions by just picking a (terminal) symbol from the expression. The system scans the set of rules associated with the symbol and attempts to unify the expression on the left side of each rule with the expression on the screen. The terminal symbol originally picked determines the context (i.e. the sub-expression) with which the unification is attempted. During the unification, the syntactical variables present in the right side of the rule are bound to sub-expressions of the expression on the screen. Each rule that unifies successfully is then displayed on the screen as an ideograph on the menu. At this point the user can select a rule to apply. The system will replace the left side of the rule with the right side in the expression on the screen using the bindings established during the unification to determine the new expression.

Besides performing all the transformation work, IDEOSY also keeps track of the user's activity in transforming expressions (i.e. in applying transformation rules) and displays on the screen the "history" of such activity in the form of a tree (the *proof tree*). Each node of the tree represents a stage in the proof reached through the transformation of an expression. Every time the user applies a new rule, the system creates a new node which is a child of the current node, and labels the arc between the two nodes with the identifier of the type of equivalence used in the transformation. An arc labeled by an equivalence symbol means that the nodes at the two extremes of the arc are equivalent with respect to that equivalence (and thus with respect to all weaker equivalences). Every node also has associated with it all the information necessary to reactivate the stage it represents. The nodes of the displayed tree are thus also ideographs that allow the user to move around in the proof by retrieving any previous stage.

5. The Interpreter

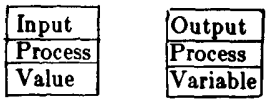
The last part of IDEOSY we describe is an environment in which CCS expressions are interpreted in an operational sense, i.e. are considered as defining actual executing modules. The interpreter subsystem consists of two main components: a *CCS interpreter* and a *user interface*. The interpreter simulates the concurrent execution of "processes" defined by CCS behavior expressions; in particular, it considers the *composition* operator of CCS as defining an execution environment in which communication channels are set up between complementary ports of processes. The user interface, based on the IDCCS graphical syntax, allows a user to interact (to "communicate") with the executing agents: the user is considered an *observer*, in the CCS sense, of the agents. In addition, the execution can be stopped at desired points and the "state" of the executing system can be displayed.

Beside being a tool for executing CCS specifications in IDEOSY, the interpreter was designed and developed for the purpose of investigating possible implementations of any language based on a port-oriented communication scheme. For this reason, the interpreter is conceived as an "abstract machine" whose structure is defined by the data-structures used in the interpreter and whose machine language is defined as a (very small) set of basic operations in terms of which the whole interpreter is implemented. These basic operations, which essentially update entries in tables, can be summarized as follows. A communication between two processes is implemented by evaluating an expression (the one being sent), updating the local environment of the receiving process and, finally, updating the "instruction pointers" of the two processes. The CCS *composition* operator is interpreted as a "create-process" primitive which provokes the allocation of new process descriptors. Private communication channels are allocated by the interpreter to implement the CCS *restriction* operator (an operator used to restrict the visibility of ports).

The user can start the interpreter at almost any point within an IDEOSY session. It will interpret the *current* expression (roughly, the expression currently displayed). Usually, the current expression is a composition of processes. The user at the "console" of the system is viewed as a process with two ports named *user-in* and *user-out*. The two ports can be used to communicate with the executing processes (i.e. to *observe* them). To do this, those processes that need to communicate with the user must have corresponding ports named *user-in* and *user-out*:

- *user-in* is used to input values from the user.
- *user-out* is used to output values to the user.

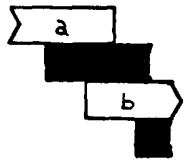
On the screen, two fields identify the two ports, as in the picture below.



The *user-in* and *user-out* actions are treated in an asymmetrical way by the interpreter. When a process tries to execute a *user-in* (*user-out*) action, the interpreter lets the user decide whether or not to "accept" the communication (communications are synchronized in CCS). A message appears on the *user-in* (*user-out*) field on the screen saying which process is trying to execute an input (output) action from (to) the user. At this point, the user

can provide a value (accept the value being output) or refuse the communication. In the second case, the action is put back in the corresponding queue in the action table and the execution proceeds with another action.

The user thus can control the execution of processes by either providing (authorizing) or not providing (not authorizing) input (output) to (from) the processes. Another tool the user can use to control the execution is a special type of action called a *checkpoint*, represented in IDCCS syntax by a black rectangle, as in the expression below.



A checkpoint is a signal for the interpreter that it must display the state of the system. Checkpoints are treated like any other action internally and inserted in a special entry always present in the action table. When the scheduler selects that entry (and when there is a checkpoint action in the corresponding queue) an IDCCS expression which represents the system at that moment is constructed and displayed. Checkpoints can be inserted in expressions anywhere an action is legal, so that the user can make the system stop and display its state at strategic instants (e.g., to check which alternative of an "or" has been selected in non-deterministic programs, right before a composition is executed to follow the creation of processes, etc.).

6. Conclusions

IDEOSY has proved to be especially interesting to us because it combines topics of theoretical interest, e.g. the study of the formalisms used for the description of programs, with topics of much more practical interest, e.g. the ergonomics of the user interface. An important test of IDEOSY will be its use in the development of a description of a large program. This test will be made as we continue the work started in [2], this time writing our description in IDCCS rather than CCS and taking advantage of the reasoning facilities provided by IDEOSY to help validate our work. As described in [2], we are investigating extensions to the semantics of CCS. One of these led to the definition of a new formalism [4] based on CCS but slightly different from CCS in the meaning of *ports*. The new formalism is still based on the basic CCS notion of agents that communicate via ports, and its constructs can be represented pictorially by the same syntax (IDCCS) developed for CCS: only the set of equational laws change, and with them the meanings of some of the ideographic constructs.

Although we do not have space here for an extensive treatment of the principles on which the pictorial syntax of IDCCS is based, it should be observed that IDCCS expressions are designed to take advantage of the possibilities offered by high-resolution screens and fast graphics hardware (for example, it is time-consuming to draw them on paper by hand). IDCCS syntax is also designed to generate expressions which are self-explanatory and which keep a recognizable shape no matter how complex they become.

The system we have described is the latest of several versions implemented over the past many months and is currently rather inefficient, due to the fact that it is implemented using a machine-independent graphics package (SGP [5]) which does not take full advantage of the hardware we are using.

We are developing several improvements for IDEOSY. One is to extend the interface so as to allow context-dependent operators. Another, which we are working on currently, is the extension of the flexibility of the user-interface, in particular with regard to the rule-definition language. Currently, the rules that can be inserted in the system can only define properties of binary operators. We are working on a generalization of an IDEOSY interface which will allow the definition of n-ary properties and will include the possibility of defining rules whose applicability depends on syntactical and semantic properties of expressions (e.g., names of ports, equality of expressions, etc.). The problem is syntactical and consists of finding a pictorial representation of indexed families of objects and conditions consistent with the rest of IDCCS and IDEOSY.

Another development of a syntactical/pictorial nature on which we are also working is giving the user the ability to define *derived* operators, i.e. new operators defined in terms of the primitive ones of CCS. The primary difficulty with this addition is finding a method by which the user may define and use new (non-trivial) ideographs without having to resort to two-dimensional parsing.

Other developments we are planning have to do with enhancing the capabilities of the system. These are in two main directions: (1) improvements to proof aiding and insertion of proof-checking capabilities in the environment, and (2) increased capabilities of the interpreter and of the editor. We plan to give the user the capability of defining sequences of rule applications ("strategies") possibly conditioned by the structure of expressions. This will make the user able to define, for example, "simplifiers" tailored on classes of expressions and will save the task of specifying every single step of expression transformations. A proof-checker will eventually be built on top of the machinery currently available to a user and will have as its main functions: (1) automating the more tedious portions of proofs, (2) providing independent verification of steps in manual proofs, and (3) "looking for" applicable strategies and proofs.

7. References

[1] Milner, R.: **A Calculus of Communicating Systems**, Springer-Verlag, Lect. Notes in Comp. Sci. 92, 1980.

[2] Doeppner, T.W., Jr., Giacalone, A.: *A Formal Definition of the UNIX Operating System*, Second ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Montreal, Canada, August 1983.

[3] Giacalone, A., Kovacs, I.D.: *IDCCS: An Ideographic Syntax for CCS*, Brown University, Dept. of Computer Science, Tech. Rep. CS-83-05, Feb. 83.

[4] Giacalone, A.: *An Approach and Some Experiments Towards the Support of Formal Specifications in Integrated Programming Environments*, Unpublished Ph.D. thesis, Brown University, April 1984.

[5] Foley, J.D., van Dam, A.: **Fundamentals of Interactive Computer Graphics**, Addison Wesley, 1982.

2008 International Conference on Computer Science and Software Engineering

Chinese language processing with complex network theory

You-Yang Yu Zhi-Qing Wang Wei-Nan Gao Guo-Qing Gu
School of Information Science and Technology, East China Normal University,
Shanghai, China
yyy82@163.com

***Abstract*—We defined two kinds of Chinese words network (CWN) in this paper. The nodes in the network are composed by the Chinese characters, phrases or classical idioms from authority dictionaries. Studying the network characters and giving a new evolution model to simulate the CWN, we found that the phrase construction follows random and preferential choosing method and we found 55 Chinese characters used by ten thousand distinct phrases. Finally, the Chinese words network exhibiting Small-world character will make it easy to search information as fast as English language.**

Keyword: complex networ; Chinese language processing

I. Introduction

Natural language is an important tool in our daily life. There are more than five thousand different languages in the world [1]. English as the most widely used language has been investigated for many years. However, using complex network theory in language study is a new method in recent years.

Since the seminal papers by Watts and Strogatz [2] on the small-world character and by Barabasi and Albert on Scale-free feature [3] were published in Nature and Science, the structure and the character of the complex network have become one of the hottest topics in physics and other relation fields. The small-world concept means two nodes connect each other in very short path but the clustering coefficient is higher than random network which has the same number of nodes. Studies have shown that in nature there are so many networks representing the feature of Small-world, such as food webs [4], human sexual contacts [5] and so on. The scale-free property, on the other hand, is defined by the power law behavior in the probability distribution of degree. The real big Scale-free network is WWW [6]. The dynamic model for Scale-free network had been proposed by Barabasi and Albert [3] which demonstrates growth and preferential attachment as two basic factors for the evolution of the Scale-free network. Although the theory is perfect and so many real networks exist, the complex network method had not been used to study the human language until 2001.

Ferrer I cancho built the first English word network in 2001[7]. He defined the node of network as the word and link as significative co-occurrences between words in the same sentence. According to the simulation, he found that the English word network is a small world network, to be specific, the average shortest path between any two words is 3. This character makes the English have higher hunt speed, which is more important for its using in World Wide Web. Conceptual network is another efficiency way to study the language using complex network. It was proposed by A.E. Motter in 2002 [8]. He constructed a conceptual network from the entries in a Thesaurus dictionary and considered two words connected if they express similar concepts. The conceptual network presenting a Small-world property indicates the information saved in our mind is efficient and is picked up rapidly.

The above successful experience proofs that the complex network theory is a good way to study natural language. In recent years, more and more people pay much attention to the network construct of Chinese language [9,10,11]. How it develops and how to use it more efficiently are the useful question deserving us to investigate.

II. Model

Chinese language has a long history. There are so many kinds of word classes, such as special words, spoken language or classical language and so on. In 2005, Li-Yong with his associates made the first Chinese phrase network [9,10] based on three different sources: daily used phrases including 10746 nodes, Internet phrase including 47951 nodes and other 96234 nodes provided by the Chinese information research group. CPN exhibits Small-word effect and power law distribution of degree as English network. But in CPN there are not Chinese characters which are more important elements in Chinese language. So we think it is useful to build a new complete network including Chinese characters and phrases.

The first important thing is to choose an authority vocabulary to make the network more practical. In our country, there are so many dictionaries. Here, we choose XinHua dictionary published by the Commercial Press in 2003 which is the first modern Chinese dictionary issued in 1953. Revised many times in large scale by hundreds of experts, this book includes almost every field terms, such as politics, economy, culture, technology and so on. Widely used in our every day life, it is a perfect word sources for Chinese language research. On the other hand, we not only study the Chinese characters and phrases but also research classical idioms, a special part in Chinese language. The classical idiom is not the same as daily idioms or proverbs. Their form is to some extent unchanged. Normally, they are composed by four Chinese characters. Classical idiom is wisdom of ancient people. The differences between the classical idioms and normal Chinese words construction may reflect

This work was supported in part by the NNSF of China under Grant No. 10635040.

978-0-7695-3336-0/08 $25.00 © 2008 IEEE
DOI 10.1109/CSSE.2008.986

language development. The source here is also XinHua idioms dictionary.

Secondly, the nodes in our Chinese words network (CWN) are Chinese characters or phrases. For any two different nodes, if one phrase contains the other phrase or Chinese character, they are linked together. For example, in figure 1, the node "计算机" includes the other four phrases or characters, i.e. "计" "算" "机 "and "计算", so it has four edges in our network.

We use $G(V,L)$ to describe Chinese words network (CWN), $V=\{v_i\}$, $L=\{l_{ij}\,|\,v_i\subset v_j \ or\ v_i\supset v_j\}$, where v_i denotes individual node i , l_{ij} denotes the connection between nodes v_i and v_j if node i contain the node j 's characters or node j contains the node i 's characters.

The difference between our Chinese words network (CWN) and CPN [9,10] is not only word source but also construction method. In CPN, the link between nodes was defined by the co-presences of the same character in the two phrases. For example, there are links between the nodes "计算机" and "机器" in CPN, but no link in CWN.

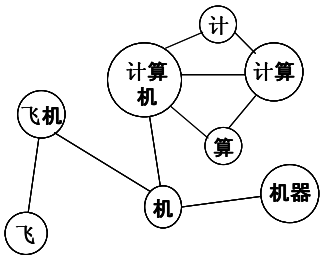


Figure 1. Network construction for CWN

Unlike English language whose basic elements for words are 26 letters, Chinese phrases are composed by Chinese characters. In Xinhua dictionary, there are 39991 distinct phrases, but only 9620 Chinese characters. How the small number of characters composes so many different phrases is an interesting question. In this paper, we try to give the answer.

III. Results and Discussion

Let us consider three important parameters in complex network, probability distribution of degree, clustering coefficient and average shortest path. Every node in network has a number of edges, which is defined as the degree k of this node. The probability of this degree k happens in the whole network is the probability distribution of the degree. For the CWN we draw the probability distribution of degree in figure 2. Figure 2(a) is for Chinese characters and phrases from Xinhua dictionary and figure 2(b) is for classical idioms. When k is small in the figure, there are the biggest value for $k\approx 7$. This phenomenon is obvious especially in figure 2(b). That is to say, when k is small, the random links in network construction is as dominant factor, but with the increase of degree k , probability distribution exhibit power law behavior satisfying $P(k)=ck^{-\gamma}$ with $\gamma=1.9$. The similitude exponent displayed by modern Chinese words network and classical idioms network shows that there exist some principle for language evolution. In order to explore the principle, we give a new evolutionary model for CWN.

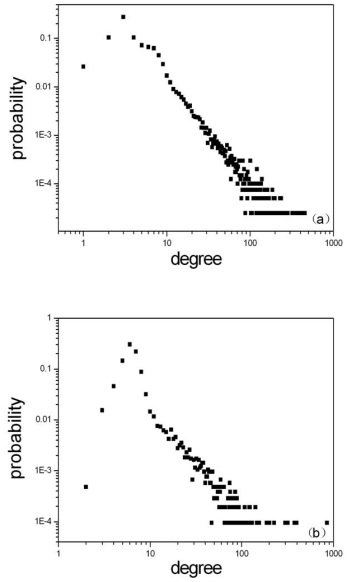


Figure 2. probability distribution of degree for CWN
(a) Chinese characters and phrases (b) classical idioms

The evolution network model (ENM) include $N+M$ nodes. There are N initialized nodes and M new nodes added to the system one by one in every time step. The generation algorithm is as follow:

(1)Initialization : N isolated points whose initialized degree is zero. The initial attract probability for every node $i\in N$ is α_0 .

(2)Grow: add a new node m to the system in each time step, until the whole new node number is M . A new node's initialized degree is k_0 , $2\le k_0<\infty$. The frequency for k_0 is $(1/2)^{k_0-1}$, recorded as $P(k_0)=(1/2)^{k_0-1}$. For example, the probability is ½ for adding a new node with degree 2 to the system, but the probability is ¼ for adding a new node with degree 3. Anyway, the bigger degree the node is with, the little probability it is added. The sum for probability satisfies formula (1). On the other hand, the initial attract probability is given for every new node $\alpha_m=\alpha_0$.

$$\sum_{M} P(k_0) = \sum_{k_0=2}^{\infty} \left(\frac{1}{2}\right)^{k_0-1} = 1 \quad (1)$$

(3) Preferential: probability Π_i that the new node will be connected to node i depends on the attract probability α_i of node i.

$$\Pi_i = \frac{\alpha_i}{\sum_j \alpha_j} \quad (2)$$

Denominator in (2) represents the sum of every node's attract probability in the system. With the increase of the degree in every node, the attract probability changed according to the following formula (3).

$$\alpha_i = \alpha_0 (1 + k_{ti}) \quad (3)$$

If the node $i \in N$, then $k_{ti} = k_i$, else if node $i \in M$, then $k_{ti} = k_i - k_0$. k_i is the degree of node i at this time and k_0 is the initialized degree for node i when it was first added to the system.

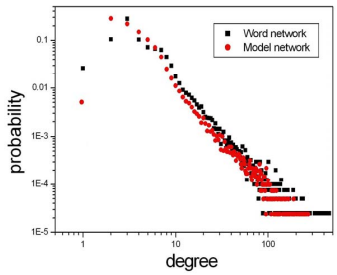


Figure 3. comparison between CWN and ENM

In the evolution rule, we found that when the node number is small, random choosing is mainly available method, but with the increase of the node, the old node with higher degree has more and more bigger attract probability, so they may become hubs nodes (with many links) in the future. The whole network's probability distribution exhibits power law behavior. Figure 3 is the comparison between the evolution network model ENM and the CWN. The experience is executed with $N = 1000$ $M = 40000$ and $\alpha_0 = 2$. The abscissa is degree of nodes, and y-axis is probability distribution. When k is big enough, the simulation error is small. So we get the conclusion that the Chinese phrases are built with preferential attachment, i.e. more widely used Chinese characters easily compose the other new phrases. We called those Chinese characters the core characters. When k is small, because of the random factors, the fitting presented relatively bigger error.

In the following, we attempt to uncover the other two statistical properties of CWN, clustering coefficient and average shortest path. In a network, let i be a node with degree k_i, it is easy to verify that there are at most $k_i(k_i - 1)/2$ edges among its k_i neighbor nodes. Let E_i be the number of the real edges existing among those k_i nodes. Then the ratio between the real and possible numbers of the edges in the sub graph of the k_i nodes is defined to be the clustering coefficient of node i, denoted C_i .namely: $C_i = 2E_i / k_i(k_i - 1)$. The average clustering coefficient for the network is $C = \sum_{i=1}^{N} C_i / N$.

The shortest path between every two nodes v_i and v_j is defined as d_{ij}. Then the average shortest path for the whole network can be written as:

$$d = \frac{1}{C_N^2} \sum_{i,j \in N}^{i \neq j} d_{ij} \quad (4)$$

By the above method, calculate the two statistical properties for CWN seeing TABLE Ⅰ. Group 1 represents Chinese characters and phrases from Xinhua dictionary. Group 2 represents classical idioms from Xinhua idiom dictionary. Because the whole network is not connective, we choose the biggest sub web to calculate those two parameters.

TABLEI. STATISTICAL PROPERTIES FOR CWN

Network	Node	Edges	C	d
Group1	35164	2059403264	0.54	2.82
Group2	10225	257642420	0.63	2.46

Obviously, CWN exhibits Small-world character, its clustering coefficient is 100 times more than the random network with the same number of nodes, but the average shortest path is the same as the random network. This result is consistent with CPN [9] although the construction method is very different from each other. Our results proved again that Chinese language has a high hunting speed as fast as English language. However, in information processing, coding high frequency using Chinese characters with relatively shorter code will save more space and raise processing speed which can not be achieved by English.

IV. Application

The degree of word in TABLEⅡ indicates the frequency in which it appeared in phases. As we know, the language is accumulated and innovated all the time. The more we are familiar with a word, the more probability we will use it make up new phases. Therefore, a word with high frequently appeared

in phases will be more important than others in daily life. There are various kinds of character sets for Chinese character, for example, GBK, GB2312, Unicode and so on. Among these sets, each character is expressed and stored by two bytes. It's really hard to modify big changes for those sets, however, on the certain condition, it can be more effective if we encode the core Chinese characters by Huffman coding. For example, in some environment, only Chinese character is accepted both in input or output, short coding length for high frequency Chinese characters will save more space and enhance processing speed.

TABLEII. CORE CHARACTER WITH ITS DEGREE IN CWN

不(454)	人(418)	大(382)	国(359)	子(332)
生(319)	地(282)	一(272)	物(271)	行(267)
心(265)	中(262)	动(261)	法(256)	义(254)
主(252)	文(248)	化(239)	无(237)	天(236)
会(235)	电(230)	体(229)	水(229)	中(262)
主(252)	学(223)	制(217)	风(215)	山(209)
海(208)	经(207)	本(200)	气(199)	马(199)
战(198)	金(198)	之(197)	分(192)	自(192)
机(186)	合(186)	合(186)	事(185)	然(184)
工(182)	业(181)	性(180)	公(177)	民(170)
成(169)	流(169)	道(168)	力(164)	平(164)

On the other hand, from the TABLE Ⅱ we can see that only one word may appear in 454 distinct phases, collecting all 55 widely used characters we will get more than ten thousand different phrases. As a new language learner, it will be efficient if he masters the core characters first. In other words, TABLE Ⅱ illustrates that a small group of Chinese characters can make up the overwhelming phases. Thus, to people who want to study Chinese fast, they can make great progress in short term if they begin from these core characters.

V. Conclusion

In our paper, we built a new Chinese words network including Chinese characters, phrases and classical idioms. This CWN exhibits Scale-free character and Small-world property. In order to explore the development of Chinese language and find out the relationship between the Chinese character and the phrase, we proposed a new evolution model which exhibits a very similar probability distribution of degree as the CWN. According to this evolution model, we found that random and preferential attachment are the most important factors in phrase construction. Although Chinese language is more complex than English, its Small-world character make it high search speed. So with the development of computer, Chinese will play an important role in information communion.

REFERENCES

[1] Hugh.Kisholl,etal., Encyclopedia Britannica, Encyclopedia Britannica, Inc, London, 1911.

[2] D.J. Watts, S.H. Strogatz, “Collective dynamics of 'small-world' networks”, Nature Vol 393, 1998, pp 440-442.

[3] A.L. Barabási, R. Albert, “Emergence of scaling in random networks”, Science, Vol 286,1999,pp.509-512

[4] Montoya. J.M.Sole, R.V., “Small world patterns in food webs”, J. Theor. Biol., Vol 214, 2002, pp:405-412

[5] Liljeros, F., Edling, C.R..Amaral, L.A.N. et al., “The web of human sexual contacts”, Nature, Vol 411, 2001,pp:907-908

[6] R. Albert, H.Jeong and A.L. Barabási, “Diameter of the world-wide web”, Nature ,Vol 401, 1999,pp.130-131

[7] Ramon Ferrer i Cancho ,R.V. Sole, “The small world of human language”, Proc, R, Soc. Lond. B, Vol 268,2001,pp. 2261-2265.

[8] A.E. Motter, Alessandro P.S. de Moura, Y.C. Lai, and P.Dasgupta, “Topology of the conceptual network of language”, Physical Review E, Vol 65,2002,pp.065102

[9] Y.Li, L.X. Wei, W.Li. Y.Niu and S.Y. Lou, “Small-world patterns in Chinese phrase networks”, Chinese Science Bulletin, Vol 50, 2005, pp 286-288.

[10] Y.Li, L.X. Wei, Y.Niu and J.X. Yin, “Structural organization and scale-free properties in Chinese Phrase Networks”, Chinese Science Bulletin, Vol 50, 2005, pp 1304-1308

[11] H.T. Liu, “The complexity of Chinese syntactic dependency networks”, Physica A, Vol 387, 2008, pp.3048-3058.

2009 International Conference on Electronic Computer Technology

A Structure Character Modeling for Chinese Character Glyph Description

Shixiao Wu, Shijue Zheng
Department of computer science
Central China Normal University
Wuhan, China
e-mail: limr2011963@yahoo.com.cn, zhengsj@mail.ccnu.edu.cn

***Abstract*—The major problem existing in current Chinese character glyph descriptions is the standard character set can't cover all possible Chinese characters, some particular variants of characters that are "unified" also can't describe and display. Character Description Language (CDL) can help resolve both of the difficulties just mentioned. This paper presents the key features and syntax of the language, and discusses some of its applications, especially to character encoding standards work and Chinese writing teaching.**

Keywords-component; Chinese character glyph description ; CDL; Character code; Chinese writing teaching

I. INTRODUCTION

For many overseas students, the most significant snag to learning Chinese is Chinese writing[1]. The number of Chinese characters contained in the Kangxi dictionary is approximately 47,035, although a large number of these are rarely used variants accumulated throughout history[2]. The quality of Chinese character is numerous, and also its structure is complicated. As one of the research interest to Chinese information processing, Chinese character glyph description offers a great convenience to overseas students who want to learn Chinese character writing.

Speaking of Chinese character glyph description, it occured us to chracter code. Character code is a mapping, often presented in tabular form, which defines a one-to-one correspondence between characters in a character repertoire and a set of nonnegative integers. Unicode & ISO 10646 is one of the well-known character codes in the world, it is a standard by the Unicode Consortium. Chinese character "一" by Unicode is U+4e00.

Character code treated the Han character as a whole, but that losed sight of its internal composition. In fact, we can decompose Chinese characters into components and strokes, that has been good for overseas students to find out the structure of Han character. Methods about Chinese character glyph description have a lot, but all in all there are five different kinds. We will elaborate that in section 2.

The problem we want to work out in Chinese charater description is the standard character set can't cover all possible Chinese characters, some particular variants of characters that are "unified" also can't describe and display. Character Description Language co-created by Tom Bishop and Richard Cook can resolve these questions. As a database language based on XML, a set of 39 strokes allow to construct a set of 1,000 components which allow to construct tens of thousands characters' descriptions. A change in the shape of one of the 39 basic strokes will be instantly visible on all other character including it[3].

we proposed the existing problem in section 1. The next part we will talk about work on Chinese character glyph description. Section 3 we expound the key feature and syntax of CDL and discuss its appliction. A Chinese character case depicited through tools called wenlindemo341 will be given in secton 4. Conclusion about our work is in part 5. References are given in section 6.

II. RELATED WORK

Work on decomposing Chinese character into components and strokes is currently led by five different ways as follows.

Han Character Information Dictionary[4], which is published by the Han character encoding group of Shanghai Jiao Tong University. It encodes essential information, includeing stroke count, stroke type, stroke order, component analysis, radicals and residual strokes, and coordinates of strokes.

Components Specification of Han character[5], providing 560 basic components as encoding units for 20902 Han characters to regroup.

IDS(Ideographic Description sequence), which describes how a Han character looks like. That is, if a specific Han character is not encoded in Unicode yet, you could use IDS to describe how the character looks li- ke. IDS consists of description characters and Han characters.

CDPL (Character Document Processing Language), a description language developed for ancient books classification, Taiwan Academia sinica proposed[6]. More than 1000 etymons is defined, over 4000 components is included.

Besides, Sun Xingming, YIN Jian-Ping, etc, proposed a novel mathemetical method to describe a Chinese character. Using this method, 505 components and 6 well-defined

978-0-7695-3559-3/09 $25.00 © 2009 IEEE
DOI 10.1109/ICECT.2009.131

operators can express all the Chinese characters successfully[7].

These five methods proposed above are all related to components, but the structure of Chinese character is complex and easily cause ambiguous. For example, “卡” can (which menns stuck) is up-down structure, it can be described by “上” and “卜”, also can be depicted “|-”and “下”. The same Character could be displayed differently by different interpreters. Also these description methods can not cover all exsiting Hanzi that are encoded and can not display particular variants of characters that are “unified”(treated as equivalent).

Then CDL can help resolve difficulties just mentioned. Compared with even the largest standard character set, CDL provides more precision: the ability to distinguish between unified variants. It also provides wider scope: a potentially infinite number of Han characters. CDL can also be used for describing and displaying characters that are not in any standard character set. The CDL instructions can be composed whenever the need arises (preferably using a graphical user interface), and included directly in a document using XML syntax.

III. CHARACTER DESCRIPTION LANGUAGE

Character Description Language (CDL) is for accurately describing and displaying the forms of all Han (CJKV) characters. This part we will presents the key features and syntax of the language.

A. The set of Basic Stroke Types

Due to the limitation of length, we only lists the set of 10 basic stroke types in Table 3.1, currently implemented in the CDL descriptions of more than40,000 ISO/IEC 10646 “CJK Unified Ideographs” are 39 types. The eleven headers A..K in Table Ⅰ are as follows[8]:

- A Sequential numbering [1..10] of all current types;
- B Numeric index for the 5 札 zhá types [1..5], with alphabetic sub-types [a..z];
- C Total number of 折 zhé ‘transitional bends’ (+1
- = number of segments) in the type;
- D Total number of control points currently implemented for the type;
- E Frequency of this type incurrent descriptions, as
- a percentage of total;
- F Glyph exemplifying the type in isolation (outside of compounds);
- G Provisional assignment of an ISO/IEC 10646& Unicode UCS (Unified Character Set) Scalar Value for each exemplar in F, or PUA (Private Use Area) for unencoded forms;
- H Name of the type in Han characters;
- I Romanization in pnyn of H;
- J Abbreviation for the pnyn name of the type in I (acronymic, except for 39);
- K Notes on the type, including structural analysis (not necessarily tied to the actual implementation), unified variants of the type, examples of usage in compounds, and cross- references to similar types.

TABLE I. THE SET OF 10 BASIC STROKE TYPES

A	B	C	D	E	F	G	H	I	J	K
1	1a	0	2	26.87	一	U+4e00	横	héng	h	horizontal;as in 大，木，三；
3	2a	0	2	15.77	丨	U+4e28	竖	shù	s	vertical;stroke 2 of 下, first stroke of 卜，stroke 3 of 丫
5	3a	0	2	12.54	丿	U+4e3f	撇	piě	p	falling to left, not very curved； as 1st stroke in 八, stroke 1 of 九
8	4a	0	2	09.59	丶	U+4e36	点	diǎn	d	taper + clockwise curve；as in 为; something to left, as 1st in 火
12	4e	1	3	00.11	乀	U+4e40	提捺	tí-nà	tn	提 tí+捺 nà; last stroke in 入,之,辶;
16	5c	1	3	03.28	𠂉	U+4e5b	**横钩**	héng-gōu	hg	一 h+left hook; 2 in 写，冗，军，农，冠，尤
17	5d	0	2	02.54	𠃌	U+200ca	竖折	shù-zhé	sz	一 h+丨 s；1st stroke in 山，or as 𠂆(丨 s+ 一 h) in 乐，东，互
21	5h	1	3	00.11	𠂇	U+21fe8	撇点	piě-diǎn	pd	丿 p+丶 d; stroke 1 in 女，妯，娌
28	5o	2	4	02.22	𰀁	U+200cc	**横折钩**	héng-zhé-gōu	hzg	一 h+丿 sg; 1st stroke in 力，2 in 月，stroke 3 of 舟
32	5s	2	5	01.84	乚	U+4e5a	**竖弯钩**	shù-wān-gōu	swg	乚sw+up hook; as in 屯，儿，心

B. *Description of A Han Character*

CDL is an XML application, which means that it conforms to a widely-used standard syntax (usage of angle brackets < >, et cetera).

Here is a description for "行":

<cdl char= "行" >
<comp char= "彳 " points="0,0 40,128" />
<comp char= "亍" points="60,12 128,128" />
</cdl>

Positions are given as points with two-dimensional coordinates. The square enclosing the entire character has (x, y) coordinates ranging from (0, 0) for the top left corner, to (128, 128) for the bottom right corner.The numbers after "彳 " describe its bounding rectangle on the left side of "行": (0, 0) is its top left corner, and (40, 128) is its bottom right corner.

In order for the above CDL description to be carried out as a set of instructions (e.g., for displaying the character or counting its strokes), it is necessary for the interpreter to refer to the separate descriptions of the components, "彳 " and "亍", as sequences of particular stroke types with specific coordinates.

Here is a description for "彳 ":

<cdl char="彳 ">
<stroke type="p" points="107,0 10,46" />
<stroke type="p" points="128,38 0,83" />
<stroke type="s" points="86,70 86,128" />
</cdl>

There are three strokes in 彳. The first two (from top to bottom) are both type 'p', which stands for 撇 piě, a curved stroke falling to the left. The third stroke is type 's', which stands for 竖 shù, a vertical falling stroke. For each of these simple stroke types, only two points are needed. For example, the first stroke starts at (107, 0) and ends at (10, 46).

C. *From Components to Strokes*

CDL uses flexible strokes to describe Chinese characters glyph. There is a form of recursion implied by CDL. For example, a description of 龍 has 16 strokes, its simplified character is 龙 [lóng] dragon. 龍 may refer (with a component) to a description of 立, which in turn may refer(with another component tag) to a description of 亠, which describes two individual strokes. A CDL interpreter will therefore typically process components within components within components, using recursive algorithms. Recursion stops when stroke elements are reached.

Any CDL description that uses comp elements can be transformed automatically into a description that uses only stroke elements. For example, 明 is described as a sequence of two components 日 and 月, each of which is in turn described as a sequence of 4 strokes. Alternatively, 明 could be described directly as a sequence of eight strokes like 3.2 we have mentioned. A straightforward recursive algorithm can transform the component description into the "strokes-only" descrition. Component descriptions are more generally useful as well as more concise.

D. *CDL and Unicode*

CDL is based on Unicode, the difference between them is the former has its own font database, the latter has Universal Character Set.

Unicode consists of a repertoire of more than 100,000 characters, a set of code charts for visual reference, an encoding methodology and set of standard character encodings, an enumeration of character properties such as upper and lower case, a set of reference data computer files, and a number of related items, such as character properties, rules for normalization, decomposition, collation, rendering and bidirectional display order.

The Unicode Consortium want to replace existing character encoding schemes with Unicode and its standard Unicode Transformation Format (UTF). Unicode can be implemented by different character encodings, the most commonly used encodings are UTF-8 (which uses 1 byte for all ASCII characters, which have the same code values as in the standard ASCII encoding, and up to 4 bytes for other characters). XML treated UTF-8 as its default character encoding form.

CDL is a character description language bulid on XML and Unicode, it provides wider scope than Unicode. By decomposing characters into strokes, CDL can cover all possible characters that existed.

IV. EXPERIMENTAL RESULT

In this section we use wenlindemo341[9] to display a hand-writing Chinese character called "土"(which means dust). First we use brush tools to write the character, just like Figure 1:

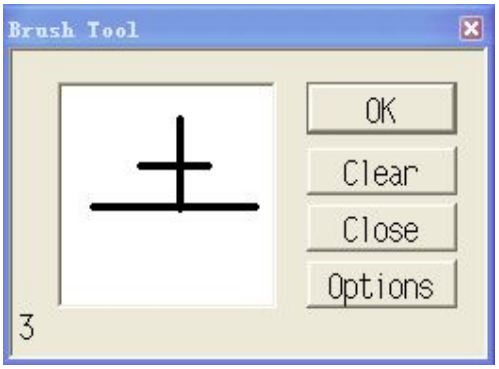


Figure 1. Using brush tools to write the character

From the bottom left corner we know this character has 3 stroke counts. Then we press ok to get lists in connection with "土" as follows:

- list characters containing 土 as a component
- list words containing 土

- list words starting with 土 (in alphabetical order)
- search files for 土
- radical 32 土
- Unicode 571f (GB cdc1) (Big5 a467)

The learners will find out "土" has 3 strokes: 一 十 土 . Considering "土" as a component, we can get characters like 在 [zài] at, 地 [dì] earth, 去 [qù] go and so on, characters containg 土 as a component have a lot. When getting wise to 土 as a radical, we have 坏 [huài] bad, 块 [huài] block, 址 [zhǐ] address and so on.

With wenlindemo341, we know character "土" can be written stroke by stroke like as Figure 2. The overseas students who believe chracter writing is painful will have fun according to use Character Description Language.

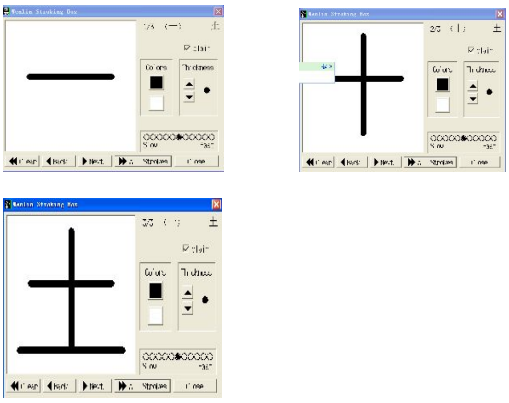


Figure 2. 1st, 2nd and 3rd stroke of 土

The wenlindemo341 has only a miniature dictionary, with a few dozen vocabulary items. The complete Wenlin dictionary has over 10,000 characters and about 200,000 compound words and phrases, including the entire ABC Chinese-English Dictionary edited by John DeFrancis.

V. CONCLUSIONS

CDL is based on Unicode, XML(Extensible Mark-up Language), and a few well-known characteristics of Han characters. It makes use of flexible strokes to display Chinese characters, maybe a helping aid to overseas students to learn Chinese characters writing.

There are many approaches to representing what might be seen as a Chinese character: bitmap, vector drawing, HanGlyph, SGML. The first two methods describe pure glyphs, whereas CDL sits somewhere between a character and a glyph, HanGlyph appears to describe almost pure haracters, but maintains some glyph-like operations that prescribe relative dimensions. SCML instead describes pure characters.

We discussed methods about character description, gived the key features and compared their differences. Decomposing a Chinese character into strokes and components is the trend of this research field, and further, application of XML will be added into these fields. All we can do is to make the character set cover all possible Chinese characters, some particular variants of characters that are "unified" can be described, traditional Chinese character and simplified chinese can both be recognized in the near future.

ACKNOWLEDGMENT

I would like to express my sincere appreciation to all the members of my team, for their encouragement and advice. Thanks to my family, they give me endless love. "The Research on Chinese Visualized Teaching Method's Model for Foreigner in Long Distance", The National Society Science Foundation of P.R. China Grant (No:2006BAK11B03) partly supported this paper."The Research on Cyberculture Security and its Warning System", Project in the National Science & Technology Pillar Program in the Eleventh Five-year Plan Period (No:07BYY033) partly supported this paper.

REFERENCES

[1] Li Xiangnong, Zhang Yi, "Design of Long-distance Visual TCFL Platform", JOURNAL OF YUNAN NORMAL UNIVERSITY(TEACHING AND RESEARCH ON CHINESE AS A FOREIGN LANGUAGE), Vol. 6, No.1, Jan., 2008.

[2] http://en.wikipedia.org/wiki/Chinese_character.

[3] Tom Bishop, Richard Cook, "A Specification for CDL Character Description Language.", 2003, pp. 2-5.

[4] 上海通大学汉字编码组. 汉字信息字典[M]. 北京：北京大学出版社，1988.

[5] 国家语言文字工作委员会.GF3001-1997 信息处理用 GB13000.1 字符集汉字部件规范[S]. 北京. 语文出版社，1997.12.1 发布，1998. 5. 1 实施。

[6] http://www.sinica.edu.tw/～cdp.

[7] SUN Xing-Ming, YIN Jian-Ping, CHEN Huo-Wang, WU Quan-Yuan, JING Xin-Hai, "ON MATHEMATICAL EXPRESSION OF A CHINESE CHARACTER", Journal of Computer Research and Development, 2002, pp.1-4.

[8] Tom Bishop, Richard Cook,. "Character Description Language (CDL): The Set of Basic CJK Unified Stroke Types",May 23, 2004, pp. 1-5.

[9] Lin Min, Song Rou., "Pattern Computing-Oriented Formal Description of Chinese Character Glyph",JOURAL of CHINESE INFORMATION PROCESSING,VOL.22, NO.3, MAY 2008, pp. 1-2.

[10] http://www.wenlin.com/cdl.

A composite approach to handle missing characters on Web interface

Chen-Yu Lai, Jan-Ming Ho, You-Qiao Wang, Zhi-Zhueng Huang

Computer System and Communication Laboratory,
Institute of Information Science,
Academia Sinica
128 Academia Rd. Sec.2, Nankang, Taipei, Taiwan 11529
{lawrence, hoho, bridge, yvb }@iis.sinica.edu.tw

Abstract. In many digital archive programs that store content, like Buddhist documents or Chinese ancient books, that contain Hanzi characters, the missing character problem is always a serious issue. Although certain solutions have been proposed to solve this problem, when it comes to Web publishing, these solutions do not comprehensively address this issue. This paper focuses on the technical approaches to display, input and search using missing characters on Web interface and use the Intelligent Character System as a example. Using these approaches, we have successfully created a system that solves the missing character problem. The interchangeability and standardization of the diverse approaches to glyph expression is beyond the scope of this paper.

1 Introduction

As more and more countries have become aware of the shift from industrial-based economies to knowledge economies, topics like Digital Libraries, Digital Archives, eGovernment and eLearning are no longer just research subjects but serious national issues. Countries have now realized that the national competitiveness will not only depend on military superiority or economic strength but, more importantly, the ability to master modern information technologies. Digital Libraries or Digital Archives, though different by definition, can effectively improve the accumulation, delivery and utilization of knowledge, and are considered essential elements of knowledge economies. The core value of information gains higher when it is been massively interchanged and shared. For many digital library programs in countries (China, Taiwan, Japan and Korea) using Hanzi characters, the missing character problem is a critical issue that not only makes compiling and reusing bibliographies less efficient but also limits the interchangeability and propagation of information.

Let's consider the missing character problem in depth. It describes a problem that people are unable to find a corresponding code point, of a specific Hanzi character or glyph, in the existing interchange code standards. It is therefore impossible to display such Hanzi characters written in ancient books or documents on present-day computer systems. It may be possible that the formal organizations of these interchange code standards never encounter that character or glyph in the character-collecting

process. Or the character or glyph is too rarely-used to be encoded during the collecting process. The major cause of this problem is that existing interchange code standards do not address the differences between characters, glyphs and variants.

A flexible approach, other than the traditional character-gathering approach like Unicode and CNS-11643/Big5+/Big5E, is to encode the missing Hanzi characters using their glyph models into a sequence composed of radicals, operators and components. The Ideographic Composition Scheme by IRG and Glyph Expression by Chinese Document Processing (CDP) laboratory in the Institute of Information Technology at Academia Sinica in Taiwan are the two methods that use this flexible approach.

2 Terminology

In this paper we will refer to several terms that are specific to Hanzi character research or Sinology research. For better understanding about the purpose of this paper, we have provieded the basic definitions for some terminology.

- **Character**: A Hanzi character is a literal unit in the Hanzi text and often represents a linguistic unit.
- **Glyph**: A glyph represents the shape or sketch of a specific Hanzi character and may have some variants, but all referring to the same character.
- **Component**: A Hanzi character can be encoded into a sequence made up of one or more sub-units. These sub-units are named as components in ICS. We use the term 'component'.
- **Root/Radical**: The roots are the subsets of components that cannot be deconstructed further.
- **Operator**: The operator is a control character that has no literal meaning but describes how the roots and components construct a glyph structure and allows to combine with other operators. Its counterpart in Unicode Standard is Ideographic Description Character.
- **Glyph expression**: An encoded sequence that is composed of operators, components and/or radicals that represents the glyph structure of a character. Its counterpart in Unicode Standard is Ideographic Description Sequence.
- **Missing character**: A missing character refers to a Hanzi ideograph that is not encoded in common character set standards and thus unable to be rendered on present-day desktop computers.

3 Basic requirements

The basic requirement for users who need to process missing characters is the ability to input, display and search using missing characters. Solutions like *Intelligent Character System* [1] and *e-Tripitaka* [2] fulfill some of the requirements. There are some common components of these two solutions. They both have an integration toolkit for inputting missing characters in office softwares like Microsoft Office, font files that contains the glyphs of missing characters and the viewer applications that can inter-

pret proprietary glyph expressions which represent missing characters. Some have the built-in capability of searching missing characters in the documents. But new problems arise when attempting to display documents containing missing characters on web interface.

4 Problem scope

These solutions (ICS, e-Tripitaka) work well on desktop computers and have been deployed in many digital archive programs in Taiwan. With the nature of the Web user interface, there are more constraints that make it more difficult to display missing characters than with the desktop user interface.

Thin Client. First, a user agent, namely a browser, is usually regarded as a thin client. That means the less prerequisite installations it requires the more user-friendly it is. With these solutions (ICS, e-Tripitaka) users are asked to acquire a copy of the software in the form of CDs or downloadable packages and install it into their personal computers. This is an annoying task even though the installation procedures of these solutions have been simplified.

Browser Compatibilities. Second, these solutions have quite different glyph expression designs and use different font packages. The *Intelligent Character System* has proprietary true-type fonts and the e-Tripitaka licenses Mojikyo Fonts from Mojikyo Font Center [4]. To display missing characters accurately in commonly used browsers like Internet Explorer, Netscape and Mozilla, the browsers must be able to recognize the glyph expressions and retrieve the corresponding font glyph correctly. This means that to display currently collected and developed font glyphs in the Web user interface, the solution providers will also need to develop browser plug-in modules that can hook the HTML page rendering procedure and make the browsers to display the missing characters. And the client-side users will be asked to install respective font files and browser plug-in modules into their operating systems and make necessary the configurations respectively.

Integrated solution. Third, current implementations do not address the requirement of offering an integrated solution for users to input keywords or terms containing missing characters and/or perform a full-text search of text materials that may contain missing characters through a browser. For now, users can perform missing character text searches on the desktop computers. Some solution providers make full-text search function available on their Web sites [2].

5 Our system design

More specifically, this integrated solution includes: a Web-based missing character input method; a Web-based missing character rendering engine that extracts the glyphs of the missing characters from the true-type font files and renders them as image files; and a glyph expression interpreter that hook the HTML rendering process

and interprets glyph expressions within the HTML source and renders the missing characters by dispatching the found glyph expressions to the rendering engine. The solution also integrates search engines that enable existing search engine systems to correctly handle text content containing missing characters. Only after seamlessly integrating these essential components into a single operating platform will content providers boost the efficient Web publishing of content containing missing characters simply, and Web users will have less painful Web experiences than they currently have.

Targeted users. Before starting the design process, there is one constraint that must first be clarified that is defining the targeted user groups. The majority of our users are historians, sinologists, archaeologists, museum officers, library officers and, of course, the Web visitors. Biologists working in Taiwan, China or Japan may also be a targeted user group. Since some species' names also contain missing Hanzi characters. These user groups share some characteristic. Most of them have basic and/or limited skills to use desktop computers and lack experience installing softwares. Therefore we summarize the priorities as, ease of use, browser compatibility and system maintainability. Fulfilling the needs of the targeted user groups was the main goal of the system's design.

Design guidelines. To draw out the most efficiency from the Web publishing of missing character content, certain guidelines based on the experiences in Web application developments should be stated before having any designs.

- The effort required to perform any prerequisite component installation on the client-side should be minimal.
- The whole design should be based on widely used technologies and standards.
- The design should be compatible to future standards and other existing standards.

Although the preliminary purpose of these guidelines is to reduce the difficulty of system deployment for the targeted academic areas in Taiwan, it is reasonable that they could also apply to system design in other countries or regions that encounter the same problem.

5.1 Web-based missing character input method editor

To edit or search missing character documents through the browser, Web users will need a Web-based interface, much like the Input Method Editors (IME), that allow Web users to compose glyph expressions of missing characters. Generally, a glyph expression is a combination of radicals, characters and escape characters [1]. However, take the *Intelligent Character System* for example, about 400 of these radicals or escape characters are user-defined characters in the Big5 user-defined area. These characters cannot be correctly entered or displayed in common desktop computers without the vendors' proprietary true-type font files installed. Therefore another requirement is to allow Web users to accurately locate the missing characters through

the browser interface in common desktop computers without installing the proprietary true-type font files.

5.2 Web-based glyph-rendering engine

This is the major component of the whole design. As stated in Section 10.1 Han of *The Unicode Standard*:

> There is no canonical description of unencoded ideographs; there is no semantic assigned to described ideographs; there is no equivalence defined for described ideographs.

Therefore it is the responsibility of the glyph-rendering engine to identify the glyph expression, locate the missing character that it represents, extract the accurate glyph from pre-composed glyphs or construct it ad hoc and display it according to the given context [5]. It is also obvious that to implement such a glyph-rendering engine will require several critical facilities: a mapping table to correlate a glyph expression with a missing character glyphholder [6]; a mechanism that differentiates between the variants of a missing character; and a collection of font files, mostly in formats such as TrueType or PostScript, that contain corresponding font styles.

The *Intelligent Character System* from CDP fulfills these requirements. As well as the glyph structure model proposed by CDP, there are now 59,766 glyphs of Kai (楷) font in the glyph database. And the glyph structure model is based on the knowledge of Hanzi. It can, therefore, be used to handle ancient fonts, such as Small Seal Script (xiaozhuan 小篆 or zhuanshu 篆書), Bronze Script (jinwen 金文) and Oracle Bone Script (jiaguwen 甲骨文). The construction of Small Seal Script is already complete. As for the Bronze Script, its construction is ongoing.

5.3 Glyph expression interpreter

For preserving missing characters in a text, the traditional approach is to mix the encoded characters along with the glyph expressions. Although this may not be appropriate for the purpose of Information Retrieval, it introduces less processing overhead than other approaches such as marking up the glyph expressions with markup languages or storing the glyph expressions in additional files. Here we provide two examples in Table 1 to demonstrate the layout of a text containing glyph expressions.

The underlined strings in the Hanzi sentences in Table 1 are glyph expressions and ideographic description sequences of Intelligent Character System and Ideographic Composition Scheme respectively. They represent 2 missing characters, 珚 and 宝. Both ⿲ and ⿳ are the horizontal conjoiner operators and ⿴ and ⿵ are the vertical conjoiner operators. Accordingly 王, 尔, 宀 and 玉 are radicals and components. In fact, these two characters 宝 (U+5B9D) and 珚 (U+73CE) are already existed in Unicode CJK Unified Ideograph but not in Big5. The Ideographic Description Sequence in Table 1 is only for demonstration purposes. In this demonstration, it is not surprising that we find a high similarity between these two ideographic structure-based approaches. However, they are now implemented, based on different character set stan-

dards, namely Big5 and Unicode. Also in this demonstration, even though we use two different ideographic structure models, they both refer to the same missing characters.

This inspired us to separate the glyph expression extraction from the glyph rendering process and, hence, make the underlying glyph-rendering engine independent of character sets. With proper object-oriented software design, this approach introduces a considerable flexibility for the compatibility with various character set standards and greatly reduces the effort that would be spent integrating various Hanzi databases.

Table 1. A missing character example

ICS Glyph Expression	拾得奇王⿰尔異宀⿰玉，如此歡喜？
Ideographic Description	拾得奇王⿱尔異宀⿱玉，如此歡喜？
Interpreted HTML	拾得奇<IMG SRC="…" ALT="王⿰尔">異 <IMG SRC="…" ALT="宀⿰玉">，如此歡喜？
Rendered text	拾得奇琢異宝，如此歡喜？

5.4 Search engine integration

Information Retrieval and Extraction also play an important role in Digital Library researches. The purpose of digitizing collections is not just to preserve them but, more importantly, to share them. Many Digital Library programs in Taiwan have built their own metadata management systems to support digitizing their collections and employed certain state-of-the-art, either academic or commercial, search engines. However these search engines may not be able to properly handle digital content containing missing characters; in other words, the glyph expressions.

6 Implementation

To implement the core elements of our system that are described in the preceding chapter, we evaluated some related information technologies.
With regard to the glyph expression interpreter, because the glyph expression identification involves many string manipulations which generally cost CPU resources, we considered applying Web server extensions like Information Internet Server Application Programming Interface (ISAPI) or Apache Module for server-side design, and Java Applet and client-side Scripting for the client-side design. For the glyph-rendering engine, we considered applying font images or Font Embedding [10].

6.1 Font images and Font Embedding

For the implementation of a Web-based glyph-rendering engine, Font Embedding, also referred to as ‘Downloadable Font’ or ‘WebFonts’, is introduced in Cascading Style Sheets level 2 [9]. This is considered to be the most appropriate solution for rendering diverse glyphs of missing characters because it is already an international

standard. However, after further evaluation of currently available Font Embedding toolkits, we decided to use the old-fashion font images instead of Font Embedding. This conclusion is a result of comparing font images and Font Embedding with several criteria in the design guidelines including the preprocessing model, bandwidth efficiency, browser compatibility and missing character interchangeability.

6.1.1 Preprocessing Model

Currently, Microsoft's implementation of Font Embedding provides the Web Embedding Fonts Tool (WEFT) to process Web pages that might require additional downloadable fonts. Before publishing these Web pages, they must be manually preprocessed using WEFT. Therefore using WEFT to handle large amount of digitized text containing missing characters is less scalable. For the preprocessing stage, WEFT analyzes the font usage on Web pages, gathers the required characters from each font used, creates the compressed font objects and then modifies the HTML page by writing in the CSS code that links the font objects to that page [10]. For the rendering stage, Internet Explorer downloads and parses HTML sources, downloads font objects referenced by that page, decompresses the font objects, temporarily installs the font objects, then renders the page [10]. Using font images, it is easy to implement the Web page conversion in different components with lots of freely available open-source imaging toolkits. Therefore the font image generation can be performed automatically without manual operations.

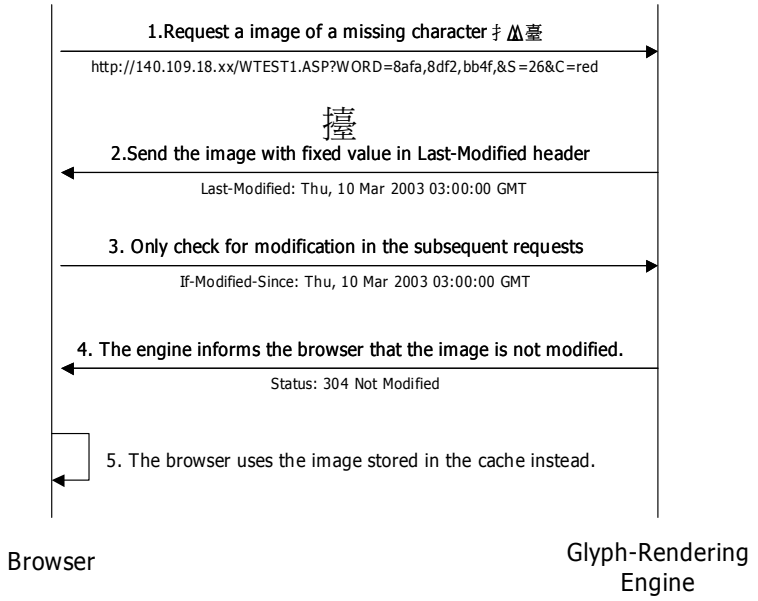
6.1.2 Bandwidth Efficiency

To evaluate the bandwidth efficiency of font images and Font Embedding, we categorized them using two criteria, the total size of downloaded data and the behavior of the cache control mechanism.

Data Size. For example take the two missing characters, 珎 and 宝. With Font Embedding, the total size of downloaded font objects is 2,431 bytes. In contrast, the size of our font images for the two characters with identical font color and font size is only 194 bytes.

Cache Control. Each time a user makes a new request to the same Web page modified by WEFT, the browser still downloads the font objects each time. But with our implementation of the font images generator that employs the Cache Control Mechanism in the HTTP protocol, the font images are only downloaded at the very first request. As shown in Fig.1, the subsequent requests are tricked by the engine to use those images stored in the browser's cache pool. Thus no redundant images are actually downloaded for subsequent requests.

Fig. 1. Cache Control in Glyph-Rendering Engine



6.1.3 Browser Compatibility

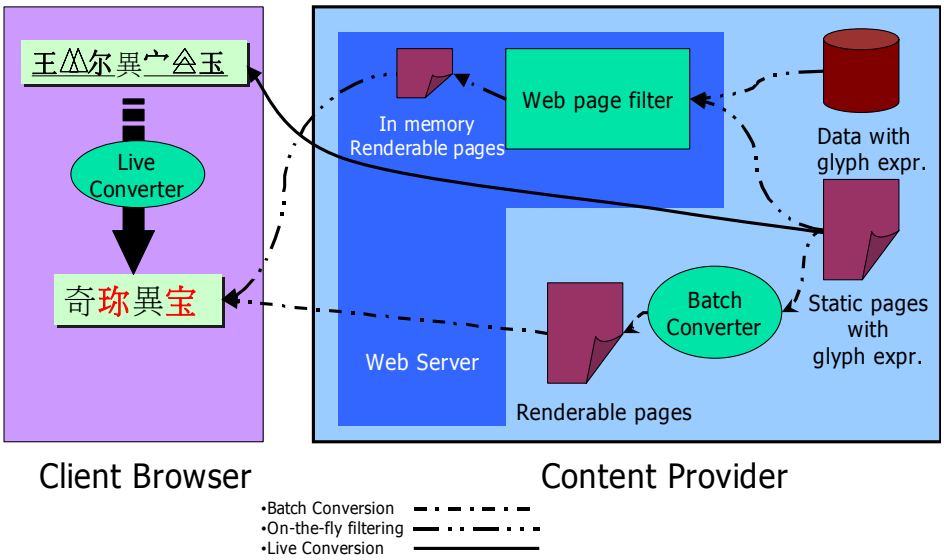
Consider again with the design guidelines in mind. Browser compatibility is not the top priority but an important index for the ease of use. The Font Embedding technology for Web interface is now only supported on Microsoft platforms. Present-day browsers other than Internet Explorer do not support the CSS notations generated by the WEFT; therefore, they are unable to render these characters decorated with the downloadable font objects in the Web pages. In contrast, with our implementation, almost every browser is able to render the glyphs of missing characters because they are provided through the use of the standard HTML tag, namely IMG. Our implementation is compatible with the following browsers: Netscape 4.x, Netscape6/7 with Java Plug-in, Mozilla with Java Plug-in and Internet Explorer.

6.1.4 Missing Character Interchangeability

As previously mentioned, the WEFT will analyze the font usages on Web pages to see what characters need to be packaged into the font objects. The common approach to display glyphs of missing characters using Font Embedding is to associate the font glyphs of the missing characters in the font files with either ordinary code points or those in the extension area for a specific character set. Therefore when the text in any Web pages is extracted either manually or by using applications, the missing characters in the text may be represented with entirely different glyphs. Using Font Embedding solves the problems of printing and display of missing characters with high-

quality results. But it doesn't address the issue of interchanging/transferring the missing characters in the Web pages to other media. In contrast to Font Embedding, with our approach shown in Table 1, after converting to missing-character-renderable Web pages, the original glyph expressions are still reserved in the ALT attribute of IMG tag. When extracting the text from these missing-character-renderable Web pages, those missing characters can also be recognized and extracted accurately, based on the hidden but readable glyph expressions. This mechanism is now implemented in the ICS applications. For example, a missing-character-renderable Web page can be loaded into Microsoft Word without losing any missing characters and the glyph expressions in the ALT attribute will be converted back to ordinary glyph expressions in the document.

Fig. 2. System configuration of our composite approach



6.2 Composite approach

Many features in our implementation are based on the practical requirements of currently developing Digital Library programs. These requirements differ in several ways namely, data sources, content formats and currently deployed facilities, either software or hardware. Accordingly we propose a composite approach to handle these various requirements.

6.3 Server-side approaches

For those organizations which have already produced or archived a large amount of Web content containing missing characters, either in the form of static Web pages or dynamic script pages, and have had powerful enough hardware support, we propose

that they use the server-side approaches. One of the server-side approaches is batch conversion and the other is on-the-fly filtering.

Batch conversion. Batch conversion is the simplest and most intuitive approach to convert Web pages containing glyph expressions into missing-character-renderable ones, and is especially suitable for static Web pages. As shown in Fig. 2, the content providers convert static Web pages into missing-character-renderable pages, which are also static pages. Then the Web servers of the content providers serve these pages to the users when they request these pages. Therefore it is considered the most effective way to render the missing characters embedded in these Web pages because theoretically the static Web pages have the best response efficiency for the browsers. However, there are still some drawbacks to this approach. First, the content will increase in time. And the content itself may sometimes need some modification that may not be reflected to the Web users immediately, due to the reasonable delay of the batch converter. Second, the batch conversion involves many string replacement operations, which often cost considerable CPU utilization. This approach is suitable for those organizations that lack adequate computing facilities to serve Web requests.

On-the-fly filtering. This is one-step-ahead improvement compared to the batch conversion. It is now implemented as a Java filter servlet mainly because the glyph expression interpreter is currently implemented as a Java component. Basically, as shown in Fig. 2, the filter servlet will perform the conversion and caching against those pages, which their URLs match the rules specified in a rule table maintained by the content provider. For subsequent requests for the same page, the filter servlet will serve the cached converted copy instead. The users of the digital libraries employing the on-the-fly filtering will benefit from the real-time reflection of the modifications to digitized content. The content providers will benefit from the separation of content publishing and content conversion because they may now focus more on content publishing or archiving. And when compared to the batch conversion, the on-the-fly filtering costs more CPU utilization that will increase with the number of Web users. Thus it will be suitable for the content providers that already have enough computing power.

6.4 Client-side approach

In contrast to server-side approaches, we propose Live Conversion as the client-side approach, which basically means that all the glyph expression interpretations are performed in the client-side browsers. We chose the Java Applet technology simply because it is widely supported by nearly all browsers. As shown in Fig. 2, the following operations are performed at the client-side browser.

- The browser sends a request to a Web page embedded with a Java applet, which is named as LiveConverter,
- The browser downloads the HTML source and the LiveConverter.
- The browser initializes the LiveConverter in the bundled Java Virtual Machine.
- The LiveConverter then downloads the page again to obtain the raw HTML source.

- The LiveConverter performs the glyph expression interpretation against the raw HTML source and caches the renderable interpreted result in memory.
- The LiveConverter renders the result on the browser window through the Live-Connect (Java-JavaScript Communication) mechanism.

With this approach, the content provider is able to provide the missing-character-renderable content easily without extensive effort. The glyph expression interpretation is completely performed at the client side thus greatly reducing the CPU utilization of the Web server. With the same facilities used in the server-side approaches, this approach will allow the Web server to serve more users.

One issue must be addressed here is the second download. The need for the second download is due to the behavior of the internal Unicode conversion functionality built in the various browsers. Basically, when the browser downloads a Web page, it will first try to convert the characters in the HTML source, encoded in any character set, to Unicode characters. This is true for Internet Explorer, Mozilla and Netscape 6.x~7with the exception of Netscape 4.x. The operators, radicals and components from CDP are coded with the code points in the Big5 extension; those from IRG are coded in CJK Unified Ideographs extensions. As a result, operators, radicals and components will be regarded as invalid characters in Mozilla, Netscape 6/7 and Netscape 4.x and will be represented as question marks '?' by these browsers. The LiveConverter is then unable to perform the glyph expression interpretation because there is no glyph expression that can be found. The second download is our solution to overcome this problem, and through the cache mechanisms implemented in the LiveConveter, we reduce the impact of the additional downloads to the Web server.

6.5 Search engine for missing character

We have now created a full-text search engine for the Web pages containing glyph expressions. After integrating the Web-based missing character IME, the search engine allows Web users to compose a query string containing missing characters and use it to perform a full-text search. And as an extended work of ICS, the search engine allows Web users to perform a full-text search against the documents generated with the ICS toolkits either in the form of Microsoft Word documents or HTML pages.

7 System deployments

There are already several deployments of this system in Taiwan. Most of them use this system to support the displaying of historical text material. Some organizations integrated this system into their internal Digital Archive management systems. The following systems are considered as mature public deployments.

- The Chinese Painting, Calligraphy, Bronze and Rubbing digitally archived at the National Palace Museum
- The Rare Book Images Search System of National Central Library

- Scripta Sinica of Academia Sinica

8 Conclusion and Future development

This system achieves its goal in reducing the efforts for displaying missing characters on Web interface. Display, input and search using the missing characters will now no longer be a technical barrier to content providers. In Taiwan, we freely provide this system for non-commercial or academic use and will contribute our source codes to the open-source community. This system now supports the *Intelligent Character System* proposed by the CDP laboratory and, if needed, can also support the *CJK Ideograph Description* easily with additional add-on implementation. Dr. Richard S. Cook from the University of California, Berkeley has submitted a proposal to add the operator characters as Ideographic Description Characters to the UCS, to the International Organization for Standardization. If the proposal is approved, it will bring more acceptance to this system. Some sinology studies propose to use markup languages like XML or SGML to represent Chinese characters. This system may also be used as a fundamental infrastructure for their studies.

9 References

1. Ching-Chun Hsieh, Chuang Der-ming : *Intelligent Character System*, A solution to Missing Character Problem, The Second China-Japan Natural Language Processing Joint Research Promotion Conference, Oct 30, 2002.
2. e-Tripitaka, Chinese Buddhist Electronic Text Association. http://www.cbeta.org
3. Ching-Chun Hsieh, C. T. Chang and Jack K. T. Huang: On the formalization of glyph in the Chinese language. A Contribution to the AFII meeting at Kyoto, Feb. 6, 1990. IIS Technical Report 1990, Institute of Information Science, Academia Sinica.
4. Mojikyo Font Center http://www.mojikyo.org/html/abroad/index_e.html
5. Christian Wittern : Some thoughts on the digitization of Kanji, http://www.kanji.zinbun.kyoto-u.ac.jp/~wittern/papers/some-thoughts.pdf
6. Ching-Chun Hsieh : A Descriptive Method for Re-engineering Hanzi Information Interchange Codes, Oct. 4, 1996.
7. A Han Unification History, Unicode Standard Version 3.0 Appendix A
8. Richard S. Cook: Proposal to add Ideographic Description Characters to the UCS, http://www.linguistics.berkeley.edu/~rscook/pdf/UniProp-Final/02221-n2480.pdf
9. Bert Bos, Håkon Wium Lie, Chris Lilley, Ian Jacobs: CSS2 Specification, W3C Recommendation 12-May-1998
10. About Font Embedding, Microsoft Developer Network Library. http://msdn.microsoft.com/library/default.asp?url=/workshop/author/fontembed/font_embed.asp

Chinese character synthesis using METAPOST

Candy L. K. Yiu, Wai Wong
Department of Computer Science,
Hong Kong Baptist University
[candyyiu,wwong]@comp.hkbu.edu.hk

Abstract

A serious problem in Chinese information exchange in this rapidly advancing Internet time is the sheer quantity of characters. Commonly used character encoding systems cannot include all characters, and often fonts do not contain all characters either. In professional and scholarly documents, these unencoded characters are quite common. This situation hinders the development of information exchange because special care has to be taken to handle these characters, such as embedding the character as an image. This paper describes our attempt towards solving the problem. Our approach utilizes the intrinsic characteristic of Chinese characters, that is, each character is formed by combining strokes and radicals. We defined a Chinese character description language named *HanGlyph*, to capture the topological relation of the strokes in a character. We are developing a Chinese Character Synthesis System CCSS which transforms *HanGlyph* descriptions into graphical representations. A large part of the CCSS is implemented in METAPOST.

1 Introduction

The rapid advancement of the Internet and the Web provides an effective means of information exchange. However, there is a very serious problem in exchanging Chinese documents: the number of Chinese characters that now exist or have ever existed is unknown. Furthermore, new characters are continually being created. Therefore, no character set can encode *all* Chinese characters.

Even if a character set could encode all Chinese characters, it is very expensive to create Chinese fonts using typical methods and a fairly large number of Chinese characters would be so rarely used that the expense would be very difficult to justify.

One possible solution to this problem is to create an unencoded character according to its composition of strokes and radicals. Several experiments along this line were attempted in the past, but none were very successful. The key reason is that the composition of the strokes and radicals is very complex, and the previous attempts did not effectively divide and resolve the complexity. The section on related works gives a brief survey of some previous attempts.

Our approach to Chinese character synthesis resolves the complexity in two ways. First, we defined a high-level Chinese character description language, *HanGlyph*. It captures the abstract and topological relation of the strokes. Thus, the character description is compact and can be targeted to a variety of rendering styles. The section on the *HanGlyph* Chinese character description language describes the language in more detail. Secondly, we use METAPOST as our rendering engine to take advantage of its meta-ness and the ability of specifying paths and solving linear equations.

The *HanGlyph* language is defined based on many studies of Chinese characters. The section on the sructure of Chinese characters explains the basic structure of Chinese characters for the benefit of readers who are not familiar with them. *HanGlyph* defines 41 basic strokes, 5 operators and a set of relations. A character is built by combining strokes using the operators recursively. *HanGlyph* allows the user to define macros to represent a stroke cluster which can then be re-used in building more complex characters.

The CCSS (stands for Chinese Character Synthesis System) takes *HanGlyph* expressions and renders the characters. It can be divided into three parts: a front-end to translate *HanGlyph* expressions into METAPOST programs, a set of primitive strokes, and a library of METAPOST macros to implement the operators and relations. By varying the parameters to these macros, or redefining the basic stroke macros, Chinese characters in different styles can be formed. Thus, it can create a variety of different fonts from the same *HanGlyph* description.

2 The structure of Chinese characters

Chinese characters, or *hanzi*, have their roots in a very long history. A large body of literature on the study of the written form of Chinese language, dating from as early as more than 2000 years ago (during the *Han* dynasty) up to now, is available. The written form of Indo-European languages consists of around 30 characters. Words are formed using these characters in a linear fashion. In contrast, written Chinese language is denoted by tens of thousands of *hanzi*. The exact number of *hanzi* that have ever existed can never be known.

Many studies have pointed out that each Chinese character is composed from strokes. The number of strokes in a character varies from one for the simplest, up to around 50 for the most complex. Unlike the linear composition of words from characters in Indo-European languages, the arrangement of the strokes in *hanzi* is two-dimensional.

According to the convention of writing Chinese characters, a stroke is a continuous movement of the brush over the writing surface without being lifted up. It is commonly agreed that there are five basic strokes: 一 (横 héng[1]), 丨 (豎 shù), 丿 (撇 piě), 乀 (捺 nà) and 丶 (點 diǎn).

In practice, each of these basic strokes has some variations depending on the position in a character. For example, the stroke 丿 撇 can have two variations: 𠂉 (平撇 pìngpiě) (as the top stroke in 千) and 丿 (豎撇 shùpiě) (as the leftmost stroke in 月). In addition, a number of combinations of these basic movements are considered as strokes because they are connected in a natural way in writing. For example, a 一 (横) followed by a 丿 (撇) is a single stroke 𠃌 called (横折撇 hèngzhépiě). Modern studies of Chinese characters [1, 9] identified a small set of around 40 strokes as the basic elements of *hanzi*.

Although the arrangements of strokes to form a *hanzi* is very complex, there are some rules that guide the formation of characters. Further, some stroke arrangements are relatively stable and appear in many characters. Some of these arrangements are themselves *hanzi*, for example, 日月; some of them are known as *radicals* which are used in Chinese dictionaries to index characters, for example, 彳匚艹. There are some relatively stable arrangements that are not *hanzi* themselves, nor radicals, but appear in many characters. We will use the term *components* to refer to all these kinds of stroke arrangements, while we use a more general term *stroke clusters* to refer to any arrangements of several strokes.

Except for a small number of very simple characters, such as 人, 二, 十, which cannot be divided into component parts, all *hanzi* can be considered as compositions of certain components. The ways of composing *hanzi* from components are known as the *structure* of the character. Many studies, such as [2] and [8], have identified around 10 different types of structures if one considers how to compose a character from only two components. This does not place serious restrictions, because the composition process can be performed recursively. Figure 1 illustrates the commonly used structures.

3 Related works

Based on the studies of Chinese characters, several attempts have been carried out to create *hanzi* from a structural composition approach.

Toshiyuki *et al* [10] proposed a way of describing Chinese characters using sub-patterns. In principle, their method is similar to the approach of *HanGlyph* because the underlying theory of character structure is intrinsic to all Chinese characters.

Dong [11] and Fan [5] reported their work on the development of a Chinese character design system which took a parametric approach to create characters in different styles. Lim and Kim [7] developed a system for designing Oriental character fonts by composing stroke elements.

Inspired by the success of METAFONT [6] in creating latin character fonts, Hobby and Gu [3] attempted to generate Chinese characters of different styles using METAFONT. A small set of strokes were defined in METAFONT. A small set of radicals were then defined as METAFONT macros by using the strokes. Characters can then be specified as METAFONT programs using these macros as building blocks. By varying some parameters governing the shapes of the strokes, fonts of different styles can be generated. However, the research was not conclusive because they only generated fonts with a very small character set (128 characters).

Another attempt similar to Hobby and Gu was done by Hosek [4] who aimed at generating *hanzi* from a small sets of components.

A common theme of the works mentioned is the difficulty of handling the complexity of the structures and the numerousness of characters. Our approach handles the complexity by using an abstract description and a layered CCSS to decompose the complexity into several sub-problems. On the *HanGlyph* level, we consider strokes as abstract objects.

[1] The word héng following the *hanzi* name of the stroke is in *pinyin*, a phonetic transcription of Chinese characters. We hope these *pinyin* transcriptions can help readers who do not know Chinese to pronounce the names of the strokes.

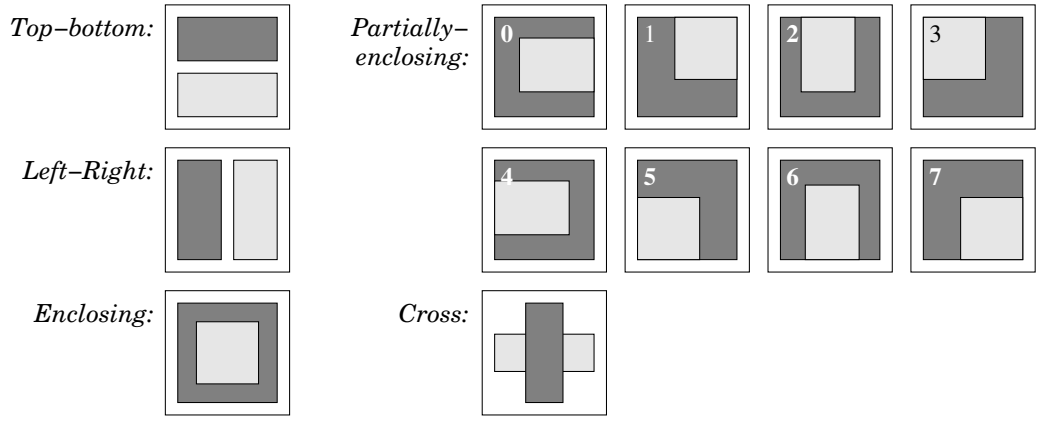


Figure 1: The basic structure of *hanzi*.

We need to specify only the relative positions between these abstract objects. On a lower level, we can work out the outline of the strokes and fine tune the positions.

Another theme of the works mentioned is that they are mainly aimed at the design and generation of character fonts. Our approach can certainly be applied in font generation. However, a very important application area, namely the exchange of Chinese character information, is made possible with our character description language *HanGlyph*.

4 The *HanGlyph* language

Based on the analysis described in previous sections, we defined a Chinese character description language, named *HanGlyph*. The most crucial characteristic of this language is that it is abstract and it captures only the topological relation of the strokes that form a character.

The essential information needed to distinguish a Chinese character is the arrangement of strokes. The precise location of each stroke can vary in a large extent up to a certain threshold, and the character can still be recognized correctly. For example, the following two characters, 土 and 士, comprise exactly the same strokes and in exactly the same arrangement. The only difference between them is the relative length of the two horizontal strokes. Exactly how much longer a horizontal stroke is in these characters is unimportant for distinguishing between them. To recognize the character 土, the threshold is that the upper horizontal stroke must be shorter than the lower one. Therefore, the *HanGlyph* language does not describe the precise geometric information of the characters.

4.1 The strokes

After studying a number of Chinese linguistic and graphological works, we selected a set of 41 strokes as the primitives of *HanGlyph*. Each primitive stroke is assigned a Latin letter as its code so that users can easily write *HanGlyph* expressions using a standard qwerty keyboard. Table 1 lists these primitive strokes.

4.2 The operators and relations

To form a Chinese character, one combines primitive strokes using operators. Five operators are defined as listed in Table 2. Figure 1 illustrates the composition performed by these operators. Each operator combines two operands to form a stroke cluster. This operation continues recursively until the desired character is formed. For example, to describe the character 土, one may first combine two horizontal strokes using the top-bottom operator, then use the cross operator to add a vertical stroke. The *HanGlyph* expression for this character (written in ASCII characters) is `h h=s+`. (*Note*: the expression is in postfix notation.)

However, with only these operators, some characters, like 土 and 士 mentioned above, cannot be distinguished. To resolve situations like this, we can augment the operator with a number of *relation specifiers* to describe the operation in more specific terms. For our sample character 土, the proper *HanGlyph* expression should be `h h=< s+_` where the symbol `<` denotes the relation that the length (i.e., the horizontal dimension) of the upper horizontal strokes must be shorter than the lower one, and the symbol `_` denotes the relation that the two operands of the cross operator, namely 二 and `|`, are aligned at the bottom.

Table 1: *HanGlyph* primitive strokes

Stroke	Name	Code	Examples	Stroke	Name	Code	Examples
丶	點	d	衣主沙	丿	撇	p	大人少
丶	左點	D	心快熱	一	平撇	P	看千毛
丶	長點	f	不	丿	豎撇	q	用月兒
丿	撇點	g	女好巡	乀	撇折	r	么絲去
一	橫	h	二三	乀	捺	n	人大丈
𠃌	橫折	i	口四國	乀	平捺	v	走趕
𠃌	橫折鉤	j	勾狗月	丿	提	t	刁打地
𠃌	橫折撇	k	又水冬	丿	點提	U	冰清
𠃌	橫折彎	l	沿船般	𠃌	橫折折	N	凹
𠃌	橫折彎鉤	m	乙吃	𠃌	橫折折折	L	凸
𠃌	橫鉤	a	冠皮軍	𠃌	橫折折鉤	J	万仍
丨	豎	s	十中用	𠃌	橫折提	E	计
𠃌	豎折	b	山區忙	𠃌	橫撇彎鉤	K	隊
𠃌	豎彎	c	四酒	𠃌	豎折折	B	鼎亞
𠃌	豎彎鉤	w	見兒	𠃌	豎折折鉤	C	馬弓
𠃌	豎提	e	衣根	𠃌	豎折撇	Q	专
𠃌	豎鉤	S	丁寸利	𠃌	橫斜鉤	M	風颱
𠃌	彎鉤	X	狗狼家	𠃌	橫折折撇	R	廷建及
𠃌	斜鉤	Y	我氣代	丿	撇點	z	学
𠃌	臥鉤	W	心感	𠃌	橫折斜	F	子魚矛
𠃌	橫豎彎鉤	o	九几				

Table 2: *HanGlyph* operators

Name	Symbol	Example
top-bottom 上下	=	昌早李
left-right 左右	|	明他你
fully enclosing 全包	@	回國困
half enclosing 半包†	^	凶問區
cross 穿插	+	十半木

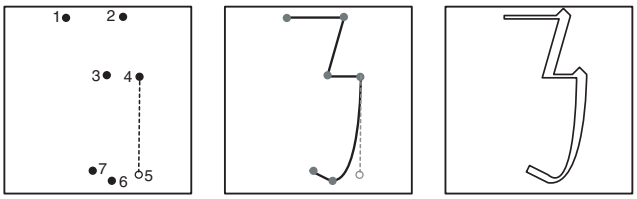
†A digit ranging from 0 to 7 should suffix the half enclose operator to indicate the direction of the opening.

The following four kinds of relations are defined:

1. Dimension — The relations in this group specify the relative dimension of the operands, i.e., comparing the width and height of their bounding boxes. There are four boolean relations: less than (<), greater than (>), not less than (!<), not greater than (!>).
2. Alignment — This specifies how the operands are aligned. The possible alignments are at top (‘), at bottom (_), at left ([), at right (]) and centered (#). More than one alignment can be added to an operation, for example, to align at bottom right (_]).
3. Touching — This specifies whether the operands can touch each other. The possible relations are touching (~) or not touching (!~). When combining two elements to form a new character, the strokes next to the interface of the two elements may or may not touch each other. In general, if the strokes on either side of the interface have the same direction, they will not touch each other, for example, 昌 晶. Otherwise, the strokes may touch each other, for example, 香相.
4. Scale (/) — This is used to adjust the width and height of the resulting character after the operation.

4.3 The *HanGlyph* macros

It would be very tedious if every character is described down to all its primitive strokes. It can be seen that certain arrangements of strokes are very common, such as 日 月, and so on. They are used to build up characters. We call them *components*. *HanGlyph* allows *macros* to be defined to stand for a component. For example, the component 日 is a macro with the name ri_4. It is defined in terms of



(a) (b) (c)

Figure 2: Basic stroke macros for the stroke 乙.

another macro `sih` representing the component 口. The actual definitions are as below:

```
let(sih){s i|/h=/}
let(ri_4){sih h@}
```

With a small number of operators and relations, and using a postfix notation, the syntax of the *HanGlyph* language is very simple. This facilitates the development of simple language processors. Figure 5.4 shows the concrete syntax of *HanGlyph*.

After defining the *HanGlyph* language, we have written descriptions of more than 3755 Chinese characters (the first level characters in the GB2312-80 character set). We found that the *HanGlyph* language is adequate for its purpose, to capture the topological relation of the strokes.

5 The CCSS

The purpose of the Chinese Character Synthesis System (CCSS) is to render the *HanGlyph* expressions into a visual representation. For example, the *HanGlyph* expression `h h =` < only specifies that there are two horizontal strokes, one above another, and the upper stroke should be shorter than the lower one. It does not tell us about the exact distance between two strokes. In addition, it does not tell us exactly how much shorter is the upper stroke than the lower one.

The task of the CCSS is to determine and calculate the precise geometric information for each stroke so that a good rendering of characters can be generated.

The CCSS consists of three modules: basic strokes, composition operations and a *HanGlyph*-to-METAPOST translator. The first two modules are implemented as METAPOST macros. The translator is a C program.

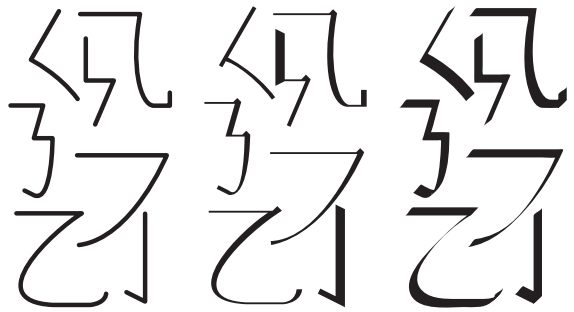
5.1 Basic strokes

Each of the 41 basic strokes listed in Table 1 is implemented as a set of three METAPOST macros:

- A *Control-point* macro defines the control points, handle points and their properties. For example, the stroke 乙 (横折折鉤 héngzhézhégōu) has six control points and a handle point as shown in Figure 2(a). The locations of the points are specified relative to a reference point. The properties of a control point indicate whether it is a beginning, an end, or a turning point. These properties will be used in creating the outline since its shape at different types of points will be different, for example, the second control point is a turning point, the outline at this point will have a serif shape. The properties will also be used in the composition operations to determine whether certain transformation and positioning opeartions are required.
- A *Skeleton* macro specifies a path passing through the control points. This path is very important. Given two points, the path can be straight or curvy; therefore, this macro traces out the exact stroke skeleton. Figure 2(b) shows the skeletal path of the stroke 乙.
- An *Outline* macro creates the outline for the stroke. It is defined relative to the control points and the skeletal path. Figure 2(c) shows the outline for the stroke 乙.

The first reason for organizing the stroke composition into three macros is to avoid distortion. In composition operations, each stroke will be transformed several times before the whole character is formed. If the stroke including the outline is represented in one macro, the transformation will distort the stroke thickness and even the direction in certain slant strokes.

Another reason is to provide meta-ness and flexibility. This organization provides several levels of style changes. The first level is to vary the parameters of the outline macros. For instance, changing the stroke thickness parameter will result in characters of varying stroke thickness. If we change the set of outline macros, we can create completely different font styles, but they may still be recognised as a family because the locations of control points are



(a) (b) (c)

Figure 3: Variations of strokes having the same skeleton.

unchanged. More variations can be achieved if the control point macros and the skeleton macros are also changed. The result will be a completely different font family. Figure 3 shows three variations of the same skeleton. Figure 3(a) is the simple skeletal path of the strokes. Figure 3(b) is the outline with serif. Figure 3(c) is generated by stroking the skeleton with a pen angled at 25 degrees.

5.2 Composition operations

CCSS implements five operations corresponding to the five operators defined in *HanGlyph*. These operations are implemented as METAPOST macros. Again, the operators in *HanGlyph* represent abstract operations, like the *Top-bottom* operator (=) only means 'put an operand on top of another'. It carries no precise geometric information. Given this abstract instruction, the macro implementing this operation has to calculate the exact location and dimension of each operand. The resulting rendering should be a well-balanced and well-positioned arrangement of strokes.

One important task of the composition operation is to estimate the relative sizes and positions for its operands so that the result is visually well-balanced. For example, Figure 4 illustrates two characters having the same radical 木 (mù) on their left side. The width of this radical in the first character 林(lín) is larger than that in the second character 樹 (shù) because the right-hand side of 樹 has many more strokes. We have found that the ratio of the widths of the two components is proportional to the ratio of the sums of the lengths of the strokes and the number of strokes of the components.

HanGlyph expressions may include a number of relations to augment the operators. The composition operations need to calculate the exact dimension and transformation to apply to each operand. For example, the character 人 (rén) is composed of

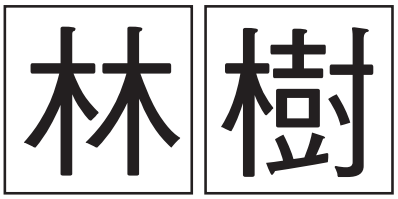


Figure 4: The same radical having different widths.

two strokes where the right-hand one is shorter and the two are aligned at the bottom. The composition operation will first scale the right-hand stroke down to a default size, and then translate it so that the bottom lines of the two strokes are aligned to the bottom of the character box as shown in Figure 5.

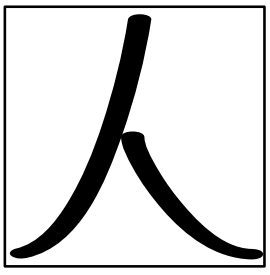


Figure 5: A character composed of two strokes aligned to bottom.

While we are talking about transforming the strokes, in fact, only the control points and the skeletal path are transformed. After all strokes forming a character have been put at the right position, the outline is drawn. This avoids the outline being distorted by the transformations.

5.3 *HanGlyph* to METAPOST translation

The front-end of the CCSS is the translator that converts *HanGlyph* expressions into METAPOST programs. The current implementation of the translator puts each *HanGlyph* expression into a METAPOST figure. Within each figure, the appropriate sequence of composition operation macros is called to render the character. The output of the system is a set of PostScript files.

This implementation provides a simple way to render the *HanGlyph* expressions and obtain previews of the characters. It facilitates the fine-tuning of the composition operations. Future implementations can streamline the process according to the requirements of the target application. For instance, a back-end processor can be added to convert the PostScript output into a particular format, such as a PostScript Type 3 font.

5.4 The syntax of *HanGlyph*

Figure 6 shows the concrete syntax of *HanGlyph* in an augmented BNF notation.

6 Conclusion

This paper describes an attempt to synthesize Chinese characters from an abstract description. A Chinese character description language known as *HanGlyph* has been defined. A Chinese character synthesis system is being developed. It is implemented in METAPOST and C, and the output is rendered in PostScript. The preliminary results show that the approach is very promising. Some of the characters generated by the CCSS are shown in Figure 7.

Currently, we are in the process of fine-tuning the composition parameters. We hope the system is able to produce visually pleasing characters. There are many factors that may affect the quality of the output, for example, the thickness of the strokes, the allocation of the space occupied by each component, and so on. Therefore, a considerable amount of experimentation is required to detemine a set of parameters for composing characters.

There are many applications of such a system. The most important ones are in exchanging Chinese textual information in an open, heterogenous environment, and in Chinese font generation.

References

[1] 蘇培成 (SU Pei Cheng). 二十世紀的現代漢字研究 (*The 20th century research on modern Chinese characters*). 書海出版社 (Su Hai Press), 2001.

[2] 劉連元 (LIU Lian Yuan). 漢字拓扑結構分析 (Analysis of the topological structure of Chinese characters). In 漢字. 上海教育出版社 (Shanghai Education Press), 1993.

[3] John Hobby and Gu Guoan. A Chinese metafont? *TUGboat*, 5(2):119–136, 1984.

[4] Don Hosek. Design of Oriental characters with METAFONT. *TUGboat*, 10(4):499–501, 1989.

[5] Fan Jiangping. Towards intelligent Chinese character design. In *Raster Imaging and Digital Typography II (RIDT91)*, pages 166–176. Cambridge University Press, 1992.

[6] Donald E. Knuth. *The METAFONTbook*. Addison-Wesley, 1986.

[7] Soon-Bum Lim and Myung-Soo Kim. Oriental character font design by a structured composition of stroke elements. *Computer-aided design*, 27(3):193–207, 1995.

[8] 傅永和 (FU Yong He). 漢字結構和構造成分的基楚研究 (*Basic research on the structure of Chinese characters and their constituents*). 上海教育出版社 (Shanghai Education Press), 1993.

[9] 傅永和 (FU Yong He). 中文信息處理 (*Chinese information processing*). 廣東教育出版社 (Guangdong Education Press), 1999.

[10] Sakai Toshiyuki, Nagao Makoto, and Terai Hidekazu. A description of Chinese characters using subpatterns. *Information Processing Society of Japan Magazine*, 10:10–14, 1970.

[11] Dong YunMei and Li Kaide. A parametric graphics approach to Chinese font design. In *Raster Imaging and Digital Typography II (RIDT91)*, pages 156–165. Cambridge University Press, 1992.

⟨*hanglyph*⟩ ::= ⟨*expr*⟩ + (1)

⟨*expr*⟩ ::= ⟨*glyph_expr*⟩ | ⟨*macro*⟩ | ⟨*char*⟩ (2)

⟨*glyph_expr*⟩ ::= ⟨*glyph*⟩`;` (3)

⟨*macro*⟩ ::= `let(`⟨*id*⟩`){`⟨*glyph*⟩`}` (4)

⟨*char*⟩ ::= `char(`⟨*code*⟩`){`⟨*glyph*⟩`}` (5)

⟨*glyph*⟩ ::= ⟨*glyph*⟩⟨*glyph*⟩⟨*opn*⟩ (6)
| ⟨*stroke*⟩
| ⟨*id*⟩

⟨*opn*⟩ ::= ⟨*parallel_operator*⟩⟨*parallel_rels*⟩ (7)
| `@`⟨*full_enc_rels*⟩
| `^`⟨*dir_all*⟩⟨*half_enc_rels*⟩
| `+`⟨*cross_rels*⟩

⟨*parallel_operator*⟩ ::= `=` | `|` (8)

⟨*dir*⟩ ::= `.(`E | S | W | N | e | s | w | n`)` (9)

⟨*dir_all*⟩ ::= ⟨*dir*⟩ | `.(`NE | SE | NW | SW | ne | se | nw | sw`)` (10)

⟨*parallel_rels*⟩ ::= ⟨*dimens*⟩?⟨*aligns*⟩?⟨*touch*⟩?⟨*scale*⟩? (11)

⟨*full_enc_rels*⟩ ::= ⟨*dimens*⟩?⟨*touch*⟩?⟨*scale*⟩? (12)

⟨*half_enc_rels*⟩ ::= ⟨*dimens*⟩?⟨*aligns*⟩?⟨*touch*⟩?⟨*scale*⟩? (13)

⟨*cross_rels*⟩ ::= ⟨*dimens*⟩?⟨*align*⟩? (14)
(⟨*align*⟩ | ⟨*intercept*⟩)?⟨*scale*⟩?

⟨*intercept*⟩ ::= `*`⟨*dir*⟩(⟨*+int*⟩(⟨*real*⟩?⟨*int*⟩?))? (15)

⟨*dimens*⟩ ::= ⟨*comp*⟩(⟨*comp*⟩ | ⟨*num*⟩)? (16)
| ⟨*num*⟩⟨*comp*⟩?
| ⟨*num*⟩`,`⟨*num*⟩

⟨*comp*⟩ ::= `<` | `>` | `!<` | `!>` | `-` (17)

⟨*aligns*⟩ ::= ⟨*align*⟩⟨*align*⟩? (18)

⟨*align*⟩ ::= `‘` | `_` | `[` | `]` | `#` (19)

⟨*touch*⟩ ::= `~`⟨*dir_spec*⟩ ∗ (20)
| `!~`(⟨*dir_spec*⟩⟨*num*⟩?) ∗

⟨*dir_spec*⟩ ::= (`.`⟨*dir*⟩) + (21)

⟨*scale*⟩ ::= `/`⟨*num*⟩? (22)

⟨*num*⟩ ::= ⟨*int*⟩ | ⟨*real*⟩ (23)

Figure 6: The syntax of *HanGlyph* descriptions.

Figure 7: Some characters generated by CCSS.

Chinese Character Recognition Based on Character Reconstruction*

Yun Li, Mei Xie
School of Electronic Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China, liyun5046@sina.com

***Abstract*—A completely new and effective algorithm for Chinese character recognition is proposed in this paper. The recognition is based on character reconstruction. And then we can obtain a new normalized character. Through character reconstruction we can reduce the error brought by the blur and the tilt of the original image. We first obtain the structure information of the original character and then we reconstruct a new character according to our rules. The structure information includes the information of the horizontal lines, the vertical line, the bias lines and dot. Then we compare the sample character and the template character and obtain the recognition result. A large number of experiments prove the robustness and high performance of the algorithm. The recognition rate of this algorithm is 96%. It can bear image's blur and tilt.**

I. INTRODUCTION

CAR license plate recognition is the key technique of intelligent traffic system. Plate recognition is widely used in traffic control, in parking lot management, in automobile surveillance. To recognize the plate license, there are several steps: you first should extract the plate from the complex background. Second transfer the image from RGB to gray and segment each character from the plate. And transfer the image from gray to binary image and then recognize each character. So recognition is the final step and it is also the most important step. Plate characters are varied from countries to countries, and number and letter characters are the most widely used. Number and letter characters can be easily recognized for the reason that they are simple. There are a large number of papers discussing about it while there are few papers discussing about the recognition of the Chinese characters appearing in Chinese license plate. In this paper we care about the Chinese character recognition. This paper proposes a completely new method to recognize them. We first get the segmented character normalized. And we compare the distance between the sample image and the template image. Of course both the sample image and the template image are the reconstructed character. Then we choose the first six characters which have the smallest distance with the sample image, and then we consider the structure information between the sample character and the chose template characters. The template which has the most similar character structure with the sample character is the final recognition result.

This paper is organized as follow. In section II, the method of obtaining isolated character is presented. The algorithm of recognition is discussed in details in section III. The result of this experiment is presented in section IV. Section V is conclusion..

II. OBTAIN ISOLATED CHARACTERS

A. Extracting Plate License

There are several ways to extract license plate from the original image [1]-[2]. Some are based on gray image and some are based on RGB image. In our algorithm we choose the method described in [1].This algorithm is based on the color space. For Chinese car plate license only have six collocations of plate's background and character. We first find the candidate location of the plate using color information and then exclude some candidate using plate's structure and texture. Then we obtain license plate and rectify it using Hough Transform. And we convert it to gray image using formula (1).

$$Gray = 0.212 * R + 0.715 * G + 0.072 * B \quad (1)$$

B. Segmentation Of The License Plate

When we get the license plate in gray, we use the method described in [3] to segment the image. Firstly, we set the standard value of the height and width of a character according to the license plate obtained. Secondly we stretch the image and use Laplacian Transform to look for character edges. Finally region growing can merge every character together and the Boundary Rectangle of the merged area with standard value of height and width is the character area. And then we can resize the isolated characters to 32*16.

C. Binarization Of The Characters

There are many methods to transfer the gray character to binary character. Some are based on self-adapt threshold, some are based on Ostu. In this algorithm we use the algorithm mentioned in [4].

III. RECOGNITION OF THE CHINESE CHARACTER

Now there are many way to recognize character [5]-[10]. To Chinese character, most scholars use SVM or BP network. They let it find character's feature by itself and get the appropriate separate line. And some scholars draw the character's features including line number information and line place information and recognize the sample character. In [8], it employs SVM to recognize both the number character

978-1-4244-4888-3/09/$25.00 ©2009 IEEE

and Chinese character. If the sample image is blurring, the result can not be so ideal. And if the image is tilt, the object pixel is not at its original place or the line width is not the same among the training set, so the predict result will not be so accurate. In our algorithm, firstly we reshape the image object to 32*16. And then we scan the whole image and get the character's structure information including the information of the horizontal lines, the vertical line, the bias line and the dot. We get their place and length and width. And now we can reconstruct the character according to the original character structure information and our rules. Finally we can recognize the sample using the reconstructed character and the reconstructed template character. During this process we have two steps. First we compare the distance between the reconstructed sample character and the reconstructed template character and choose the first six template characters with the smallest distance. And then we use some structure information to get the final recognition result.

A. Extract The Boundary Rectangle Of The Object And Resize It

The object is not always occupy the whole image, some are in the upper part of the image, some are in the lower part of the image and some are in the middle. To make the line place information more accurate and accordant, we extract the contour of the image object and resize it to 32*16. Thus we can eliminate the error cased by different size of objects to some extent. We scan the image row by row and then can obtain the first row index of non zeros and the last row index of non zeros. This row indexes are the contour's upper limit and lower limit. When we scan the image along the column using the same method, we can get the contour's left limit and right limit. Then we resize this rectangle to 32*16 using bilinear interpolation.

B. Drawing Character's Structure Information

When we have processed the image according to step 1, then we can say the image object occupy the whole image. So we can diminish error to some extent.

There are four kinds of structure information we will draw. First we must know the character's structure. There are four kinds of character structure in Chinese: up-down structure, left-right structure, and the whole structure and half-encircled structure. But we suppose Chinese character has the first three kinds of structure. The second structure information is the information of the horizontal lines and vertical lines. The third structure information is the information of the bias lines. And the fourth structure information is the information of the dot. In our algorithm, we adopt a simple way to decide character's structure. We scan the image and get the longest horizontal line and the longest vertical line. The character's structure is decided as formula (2).

$$structure = \begin{cases} up-down & if: l1/c > 0.8 \,\&\, l2/r < 0.8 \\ left-right & if: l2/r > 0.8 \,\&\, l1/c < 0.8 \\ whole & if: l1/c > 0.8 \,\&\, l2/r > 0.8 \end{cases} \quad (2)$$

In the formula, l1 is the length of the longest horizontal line, l2 is the length of the longest vertical, c is the column number of the image and r is the row number of the image.

Here we label the image first using 4-connection and dispose the area one by one. We first decide whether the area is a dot. If the length and the width of the area are similar and they are less than a threshold which is set at 5 pixels and the distance of them is less than a threshold which is set at 3 pixels then we believe it is a dot and we record their information. If an area is a dot then we continue to scan the next area and record our structure information. If the area has no dot then we continue to search for horizontal line and vertical line.

Now we began to scan the area to get our line information. Searching for horizontal line and vertical line has the same algorithm. For the reason that some image is tilt and so the horizontal line is not so horizontal and the vertical line is not so vertical, so when searching for the lines we must take it into consideration. And for the reason of binaryzation, some pixels in the line have value of 0, and the algorithm must tackle it. We suppose that the line length is the number that have longest continuous pixel with value one. And if the line length we obtained is less than a threshold which is decided by the actual image and then we believe it should be noises or part of a vertical line. When searching for horizontal line we discard it. When we count one continuous pixels length we use algorithm as follow:

$$\begin{aligned} &continuoun\quad umber\quad = continuous\quad number\quad + 1 \\ &if\ : image\ (i, j) = 1\ \&\ image\ ((i + m), (j + 1)) = 1 \end{aligned} \quad (3)$$

Here we suppose we reach pixel image(i, j) and its pixel value is one and when we search for the next pixel we will take pixel image((i+m),(j+1)) into consideration. Here m is a measure of tilt angle. It is determined by your image. We measure image's tilt angle by pixel number. When the line is completely horizontal its tilt angle is 0 pixel. If your image's tilt angle is 0 pixel then we give m value 0. The value of m is in proportion to image's tilt angle. In our algorithm the value of m is 1. And finally we can obtain the information of the line including the row index, the column index, the length, and the width.

To obtain the information of the bias line, we scan the whole image. We first look for a seed. If we search for left bias line, the seed should satisfy the condition that its pixel value is 1 and its left pixel has value of 0. And we continue to search left bias line at the direction of -135 degree. We can search for our right bias line using the same method.

C. *Reconstruct Characters*

So far we have got all the information we wanted. The following thing we will do is to reconstruct our character.

According to our statistics, we suppose the width of the horizontal line and the vertical line is 2 pixels. If the width of one line is more than 5 pixels, then we believe two or more line is stuck each other. Then we divided them into two or more lines base on rules that the width of line is 2 pixels and the interval of adjacent lines is 2 pixels. And most of the characters have no more than five horizontal lines and four vertical lines. So we suppose all of the characters have no more than five horizontal lines and four vertical lines. And both the horizontal line and the vertical line are located in the permanent place. To our image of 32*16, we suppose all horizontal lines are located in row 3, row 11, row 16, row 23 and row 28 while all vertical lines are located in column 3, column 8, column 12, and column 15. If one horizontal line is not in the extra row then we suppose it is cause by image preprocessing or noises. And we will adjust the row place of the line to the nearest horizontal line's place determined by our hypothesis. We suppose the beginning and the ending of one horizontal line are at one vertical line's column place. If the beginning or ending column index of one horizontal line is not at the given column, then we will also adjust them to the nearest vertical line's column place determined by our hypothesis. Thus we complete the reconstruction of horizontal line. We can reconstruct vertical line using the same method.

Now we will reconstruct the dot and the bias line. According to our dot information, we know the place of the dot. We may obtain many dots, but some of the dots may be noises. So we must tell them. We can do this from the statistics of dot place. If the place of the dot may exist a dot, then we consider the dot as part of the character otherwise we consider it as noises. Then we reconstruct the dot at the extra place with permanent size. With the same method we can reconstruct bias line. But when we reconstruct the bias line we reconstruct the bias line as a horizontal line. So we can reduce error caused by the fact that some horizontal line may be judged as a bias line.

And now we finish reconstructing one character. Fig. 1. is parts of characters of the original templates and their reconstruct characters.

Fig. 1 Part of templates and their reconstructed characters

D. *Recognize The Sample*

Now we employ template matching to recognize sample character. Both the template character and the sample character are the reconstructed character. So they are more standard, and the error caused by image tilt, image noises, and some other causes is reduced, however, the reconstruct character can not replace the original character for the reason that they loss some information in standardization. We will recognize our sample character through two steps. Firstly we transform the character matrix into vector along column, and then calculate the Euclidean distance between the sample vector and the template vector and get the six candidates that own the minimal distance. Secondly we use the structure information to get the final recognition. We will describe how we use the structure information to obtain the final recognition result in details.

Although the reconstruct characters are more standard, they loss part of information. So the result obtained through vector matching of reconstruct character is not so reliable, but the correct result is always in the first six candidates. So we choose the first six candidates to carry out our second recognition. We dispose them as follow: we compare the structure information between the sample and the template, and the template which own the most similar structure information are our recognition result

IV. EXPERIMENT RESULTS

In our experiment, we use real 24-bit 800*600 color images captured at varied background including parking lot and freeway and some other places. According to the Standard for Vehicle License Plate in the People's Republic of china, there are about 60 Chinese characters and 34 character of the abbreviation of province. In our experiment, we recognize the 34 character. We employ 800 numeric and alphabetic images. In following table, we compare our method with SVM method mentioned in [8]. The traditional template matching result is reported in [9].

TABLE I ACCURACY COMPARISON

Methods	Character
SVM	95.7%
Traditional template matching	92.7%
Our method	96%

Fig. 2. Parts of sample characters

From the table we can conclude that our algorithm have slightly advantage over SVM method and is better than traditional template matching method. Our algorithm recognizes character through structure information so our algorithm can recognize it only if the character's structure is complete. Our algorithm can tolerate the adherence of the image and can tolerate the tilt of the image. So on the whole our algorithm has advantage. Figure 2 the part of our test sample image.

V. CONCLUSIONS

In this paper we present a complete new method to recognize the Chinese character in License plate. Our recognition is based on Chinese character reconstruction. And we use the reconstructed character and its structure information to obtain recognition result. The algorithm performs well on recognizing character which has complete structure. And our algorithm can tolerate the adherence of the image and the tilt of the image. Our algorithm perform well on robustness.

REFERENCES

[1] Yao-Quan Yang, Jie Bai, Rui-Li Tian, Na Liu, "A Novel Binarization Approach for License Plate," 2006 1ST IEEE Conference on Industrial Electronics and Applications, May 2006, pp. 1-4

[2] Feng Yang, Zheng Ma, "Vehicle License Plate location Based on Histogramming and Mathematical Morphology," 2005 Fourth IEEE Workshop on Automatic Identification Advanced Technologies, 17-18 Oct 2005, pp:89-94.

[3] Feng Yang , Zheng Ma , Mei Xie, "A Novel Approach for License Plate Character Segmentation," 2006 1st IEEE Conference on Industrial Electronics and Applications, May 2006, pp. 1-6

[4] Feng Yang , Zheng Ma , Mei Xie, "A Novel Binarization Approach for License Plate," 2006 1st IEEE Conference on Industrial Electronics and Applications, May 2006, pp. 1-4

[5] Yo-Ping Huang, Shi-Yong Lai and Wei-Po Chuang, "A Template-Based Model for License Plate Recognition," 2004 IEEE International Conference on Networking,Sensing and Control, Vol.2, 2004, pp.737-742

[6] Ratree Juntanasub , Nidapan Sureerattanan, "Car License Plate Recognition through Hausdorff Distance Technique," Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence, Nov 2005, pp. 5, 14-16

[7] Xiaojun Chil, Junyu Dong , Aihua Liu, Huiyu Zhou, "A Simple Method for Chinese License Plate Recognition Based on Support Vector Machine," Proceedings of 2006 International Conference on Communications,Circuits and Systems, Vol 3, 25-28 June 2006, pp:2141-2145

[8] Lu Xiaobo, Ling Xiaojing, Huang Wei, "VEHICLE LICENSE PLATE CHARACTER RECOGNITION," Proceedings of the 2003 International Conference on Neural Networks and signal Processing, 14-17 Dec 2003, pp.1066-1069, Vol 2

[9] Siti Norul Huda Sheikh Abdullah,Marzuki Khalid and Rubiyah Yusof "License Plate Recognition using Multi-cluster and Multilayer Neural Networks," 2006 ITTC 2nd Information and Communication Technologies, Vol 1, 24-28 April 2006, pp.1818-1823

Resolving the Unencoded Character Problem for Chinese Digital Libraries

Derming Juang, Jenq-Haur Wang, Chen-Yu Lai, Ching-Chun Hsieh,
Lee-Feng Chien, and Jan-Ming Ho
Institute of Information Science, Academia Sinica, Taiwan
{derming, jhwang, lawrence, hsieh, lfchien, hoho}@iis.sinica.edu.tw

ABSTRACT
Constructing a Chinese digital library, especially for a historical article archiving, is often bothered by the small character sets supported by the current computer systems. This paper is aimed at resolving the unencoded character problem with a practical and composite approach for Chinese digital libraries. The proposed approach consists of the glyph expression model, the glyph structure database, and supporting tools. With this approach, the following problems can be resolved. First, the extensibility of Chinese characters can be preserved. Second, it would be as easy to generate, input, display, and search unencoded characters as existing ones. Third, it is compatible with existing encoding schemes that most computers use.

This approach has been utilized by organizations and projects in various application domains including archeology, linguistics, ancient texts, calligraphy and paintings, and stone and bronze rubbings. For example, in Academia Sinica, a very large full-text database of ancient texts called Scripta Sinica has been created using this approach. The Union Catalog of National Digital Archives Project (NDAP) dealt with the unencoded characters encountered when merging the metadata of 12 different thematic domains from various organizations. Also, in Bronze Inscriptions Research Team (BIRT) of Academia Sinica, 3,459 Bronze Inscriptions were added, which is very helpful to the education and research in historic linguistics.

Categories and Subject Descriptors
H.3.7 [**Information Storage and Retrieval**]: Digital Libraries; H.3.5 [**Information Storage and Retrieval**]: Online Information Services.

General Terms: Languages

Keywords
Character Encoding, Unencoded Chinese Characters, Digital Library, Glyph Expression

JCDL'05, June 7–11, 2005, Denver, Colorado, USA

1. INTRODUCTION
Missing or unencoded characters for a language present practical problems in digital library applications including character encoding, text input, font generation and display, document distribution and searching, and so on. The problems are particularly serious for large character sets, such as the Chinese character[1] set, and other Han-related character sets. Constructing a Chinese digital library, especially for a historical article archiving, is often bothered by the small character sets supported by the current computer systems. This paper is aimed at resolving the unencoded character problem with a practical and composite approach for Chinese digital libraries.

In Chinese digital libraries, there's a fundamental problem in representing unencoded Chinese characters. The number of Chinese characters has been growing over time. This is due to the extensibility of Chinese characters which comes from the componential nature of characters. Moreover, the set of Chinese characters is not a closed finite set, and there's a lack of character standard.

The number of Chinese characters far exceeds the available coding space in computers. For example, the GB-2312 code standard contains 6,763 characters in Simplified Chinese, while the Big5 code standard contains 13,053 characters in Traditional Chinese. Even though the Unicode Standard [20] tries to solve the problem by expanding the coding space to a much larger space of 70,207 characters in Unicode 3.1, the fundamental problem still exists: there're always some Chinese characters unencoded. According to a statistics in a full-text database for Chinese ancient documents called Scripta Sinica [19], there are 22,780 distinct characters in a total of 200 millions characters of ancient texts such as Ershiwu shi[2], Shisan jing[3], … etc. Among these 22,780 characters, 9,727 (42.7%) [11] cannot be encoded using Big5 code. Even in modern usage of Chinese characters, the problem still exists. For example, some special characters in person names and location names cannot be correctly displayed and input. Due to naming convention or personal preferences of characters, people can always coin new characters according to their needs. In fact, there are more than 18,000 unencoded Chinese characters in the person names of the

[1] In this paper, we use the term "Chinese characters" to represent the CJKV ideographs or Han-related characters (漢字) which are commonly referred to as Hanzi in Chinese, Kanji in Japanese, or Hanja in Korean.

[2] Ershiwu shi 二十五史, The Twenty-five Dynastic Histories.

[3] Shisan jing 十三經, The Thirteen Confucian Classics.

Household Registration in Taiwan. More than 10 new characters are being added per month.

Research of Chinese digital libraries started in the mid-1990s. In these digital libraries, various full-text databases of ancient and modern documents has been built. For example, China Digital Library Project (CDLP) [13, 24] in China began in year 2000 with a 5-year timeframe building contents and infrastructures based on global standards. National Digital Archives Project (NDAP) [17] in Taiwan aims to promote and coordinate content digitization and preservation at leading museums, archives, universities, research institutes, and other content holders. During these efforts, four aspects of impact on these digital libraries were observed, including character encoding, text input, font generation and display, and document distribution and searching. First, in small-scale solutions, a "user-defined font area" in the coding space has been commonly adopted for users to specify new code sequences and the corresponding fonts for unencoded characters. However, the limited user space, the tedious font creation process, and the problem of code sequence ambiguity could make the digital library contents unable to be shared. Second, it would be very difficult to memorize and input the code sequences. Third, fonts have to be created by users and displayed when a user area code is detected. Finally, searching for such characters is difficult if not impossible. Therefore, a total solution for the unencoded Chinese character problem is required.

To resolve the unencoded character problem, there are three requirements in computer representation. First, the extensibility of Chinese characters has to be preserved. Second, it should be as easy to generate, input, display, and search unencoded characters as existing ones. Third, it must be compatible with existing encoding schemes that most computers use.

In this paper, a composite approach to the representation of Chinese characters is proposed. The proposed approach consists of the glyph expression model, the glyph structure database, and supporting tools. As of the writing of this paper, the glyph structure database contains 59,220 characters of Standard Scripts with 12,681 sets of variants. Other characters include 177 Oracle Bone Inscriptions (甲骨文) with 25 variants, 3,459 Bronze Inscriptions (金文) with 1,055 variants, 11,100 Small Seal Inscriptions (小篆) with 1,081 variants, and 372 Chu Bamboo Slips Characters (楚系文字) with 73 variants. More characters are continuously added. The goal of this approach is to propose a complete solution that can fully express Chinese characters including existing ones, both ancient and modern, and those to be coined. All these characters can be input, displayed, distributed, processed, and even searched in exactly the same way as ordinary characters without introducing too much inconvenience to users.

This approach has been utilized in various applications. First, in Academia Sinica, a full-text database of ancient texts called Scripta Sinica [19] was created which contains a very large collection of historic documents such as Ershiwu shi, Shisan jing, … etc. Second, the Union Catalog of NDAP [21] dealt with the problem encountered when merging the metadata of 12 different thematic domains from various organizations. Third, in Bronze Inscriptions Research Team (BIRT) of Academia Sinica, 3,459 Bronze Inscriptions were added, which is very helpful to the education and research in historic linguistics. More systems in museums, archives, and content holders are being built using the proposed approach.

2. BACKGROUND AND RELATED WORK

In order to make for a more comprehensive reading for people not familiar with Chinese characters, an introduction to the background knowledge about Chinese characters, glyphs, encoding standards, and digital library efforts is provided.

2.1 Chinese Characters and Glyphs

The Chinese writing system was originally pictographic, that is, words were represented by pictures that closely resembled the meaning of the word. For example, the picture for "sun" looked much like the sun. This pictographic writing eventually developed into the more complex ideographic writing that we are more familiar with. Chinese writing is one of the only contemporary writing systems that still prominently bear traces of its pictographic origins.

Around 1500BC, Chinese scripts started to appear in inscriptions. The first national standardization effort of Chinese scripts waas done in the realm of the First Qin Emperor (c. 259-210BC). Several lexicons were built since then. Around 121AD, in the Eastern Han Dynasty, Shuowenjiezi[4] tried to analyze and classify all of the 10,516 characters at that time. In the 12th century, Guangyun[5] collected 25,126 characters. In 1716AD, Kangxi Dictionary[6] contained 47,043 characters. The number of Chinese character was increasing over time.

Chinese characters are differentiated only by the meanings they carried [6]. A character may have many *glyphs* or shapes. Glyphs are differentiated by their structures or skeletons. For example, the characters 'sun' and 'moon' have different meanings. The glyph of 'moon' in Standard Scripts looks quite different from the same character in Bone Oracle Inscriptions. During the history of Chinese characters, many different forms have been derived, for example, Oracle Bone Inscriptions (甲骨文), Bronze Inscriptions (金文), Seal Inscriptions (篆), Official Script (隸書), and Standard Script (楷書). The evolution of characters can help us trace the origin and better understand their meanings.

Glyphs do not deal with the actual outlook of the writing, but fonts do. A font is a collection of general design rules for a specific style of typefaces. For example, it's possible to specify the size, the broadness of strokes, the ratio of broadness of vertical and horizontal strokes, etc. For the same glyph, there are different styles of writing and calligraphic variants such as 宋 (Song), 隸 (Clerical Scripts), 行 (Running Scripts), 草 (Grass Scripts), etc..

Chinese characters are extensible or productive in the sense that new characters can be generated out of existing ones. It's fundamental since a finite set of basic ideographic components and composition operators can be used to express an unlimited set of unique Chinese characters. There are two levels of elements in Chinese characters: *strokes* and *components*.

[4] Shuowenjiezi 說文解字, "Elucidations of the Signs and Explications of the Graphs," compiled by Xu Shen (許慎), 100AD.

[5] Guangyun 廣韻, an ancient Chinese rhyme dictionary.

[6] Kangxi Dictionary 康熙字典, a dictionary compiled by Zhang Yushu (張玉書) et. al., 1716AD.

Strokes are the atomic element constituting the writing and calligraphic shape in Chinese characters. From a serial group of strokes, components can be formed, which are the basic compositional units for characters. There are two types of components: *basic* and *compound*. Basic components, also known as *radicals*, are the minimum fundamental components that will not be further decomposed. Compound components are intermediate elements consisting of several radicals. In order to uniquely represent a glyph, it's natural to decompose a glyph in terms of its radicals.

In 1972, Ni [18] analyzed 16 different compositional operators of Chinese glyphs and found that only three of them, namely horizontal composition, vertical composition and containing composition, are frequently used and can be assembled as an effective system for representing the structure of glyphs. After analyzing the 9,129 glyphs in [12], 496 radicals were obtained. Named as the NCTU Radical Set (a.k.a Chiao-Tung Root System), it was the earliest study of glyph structure in Taiwan.

This radical set is unique in that it was obtained by optimizing the total number of the radicals and the averaged number of radicals per glyph. In general, the less the number of radicals, the longer the decomposition of glyph. In this set, the weighted average number of radicals per glyph is only 1.9. After checking against the 49,905 characters of a Chinese lexicon[7], 48,713 characters (97.6%) can be expressed using this set. The remaining 1,192 characters are ancient scripts. More details of the statistics and optimization of the NCTU Radical Set can be found in [9, 12].

2.2 Chinese Character Encoding Schemes

In order to represent Chinese characters in computers, many different encoding schemes have been proposed to distinguish among characters. For example, each character can be uniquely assigned a 2-byte code in schemes such as GB-2312, Big5, …, etc. However, the encoding standard for Chinese characters is still in lack. In China, the Simplified Chinese encoded in GB-2312 code standard contains 6,763 characters, while GBK code contains 20,902 characters. In Taiwan and Hong Kong, the Traditional Chinese in Big5 code standard contains 13,053 characters, while the CNS 11643 code standard contains 48,027 characters. The Unicode Standard 3.1 [20] tried to unify all CJKV ideographs with a set of 70,207 characters.

Since there are more than 65,536 unique Chinese characters, only the frequently used characters can be represented if each character is represented by one 2-byte code. In existing encoding standards such as GB-2312 and Big5, no extra information such as the structural evolution and variations of characters is encoded except the code sequence of glyphs, which has no particular meaning besides indexing. In the character-glyph model [20], the underlying model in Unicode, abstract characters with semantic meanings are differentiated from concrete glyphs, the visual shapes of characters. Only characters are encoded, not glyphs [3]. Characters from existing CJKV character sets must be merged, not separately encoded. Therefore, the Unihan database [10], the largest subset of the Unicode Standard, was introduced as the repository for the Unicode Consortium's knowledge regarding CJKV ideographs.

[7] Zhongwen da cidian 中文大辭典, "Great Dictionary of the Chinese Language," edited by Zhang Qiyun, 40 vols., 1962-1968. Revised edition, 1973, 10 vols.

Different from the Unicode model, we adopt an analytic representation of characters, by breaking them down into pieces or components. This is similar to an ideographic composition scheme called Ideographic Description Sequence (IDS) [14, 15] later adopted by Unicode with a different set of operators and components. Recently, a character description language (CDL) [1] was also proposed to describe the characteristics of characters, which is more versatile than IDS.

2.3 International Efforts in Chinese Digital Libraries

Many international efforts dealt with the problems encountered when building Chinese digital libraries. Wittern [22, 23] introduced the problems encountered when dealing with digitization of Chinese Buddhist texts. CHANT project [4] in Hong Kong aimed to build an electronic database of all 1500BC to 600AD Chinese ancient texts. Richard Cook [2] discussed the character set issues relating to the computerization of an ancient Chinese lexicon Shuowenjiezi.

In China, CDLP (China Digital Library Project) [13, 24] was a 5-year project starting from year 2000 building contents and infrastructures based on international standards. In Taiwan, NDAP (National Digital Archives Project) [17] started from 2002. Based on the experiences obtained from Digital Museum Project, Archives Digitization Project, and International Digital Library Cooperation Project, NDAP aims to digitize and preserve all kinds of documents across various organizations in different domains.

2.4 A Brief History on Handling the Unencoded Characters

Before the prevalence of personal computers, there has been some search on building a Chinese computer which is capable of inherently representing, encoding, inputting, and displaying Chinese characters. Since the number of Chinese characters far exceeds the alphabets in English, it's impractical to put each character in a separate key. In fact, in Chinese publishing industry, there was a "big" keyboard consisting of some 5,000 frequently used characters. A smaller version of some 500-key keyboard was also adopted.

The preliminary research on the problem of unencoded characters was based on the study in Chinese characters for computer use in National Chiao-Tung University (NCTU) [12, 18]. The fundamental set of radicals and character composition operators was analyzed.

In 1984, the needs for the digitization and preservation of Chinese ancient texts grew stronger in Academia Sinica, and solutions on the related issues of Chinese character encoding and glyph rendering started to take shape [5, 7, 8, 11]. Digitized ancient texts originally included Ershiwushi, which later evolved into Scripta Sinica [19] consisting of more than 200 million characters in total. In 1993, more than 9,700 unencoded characters have been collected. We started to focus on the solution on glyph expression model and the glyph database, which also evolved over the years.

Since 2002, NDAP [17] was started and the unencoded character problem became a critical issue when digitizing, organizing, and integrating more diverse texts and multimedia objects from various domains. Up to now, more than 100 databases from 12 thematic groups were integrated using the OAI-based [16] Union Catalog [21].

With growing popularity and usage of the Web, our PC-based solution was improved to accommodate the increasing needs for Web documents processing. Web-based intelligent character system

was proposed to deal with the input, display, and search of unencoded characters in Web pages. A Web-based input method was used to compose unencoded characters. A browser plug-in module was designed to filter the Web pages, convert the glyph expressions into corresponding glyphs, and display them on the fly.

3. GLYPH EXPRESSION AND GLYPH STRUCTURE DATABASE

In the proposed approach, there are three main components, the glyph expression, the glyph structure database, and supporting tools.

3.1 Representation of Chinese Glyphs

Our glyph model is shown in Table 1. In this model, there are two parts: the representation of glyphs and the glyph structure expression. A formal representation of Chinese glyphs in Backus Naur Form is shown in Table 1.

Table 1. A formal representation of Chinese glyphs in Backus Naur Form.

<character set> ::=	<glyph> | <symbols>
<symbols> ::=	punctuation symbols, pronunciation symbols, and others
<glyph> ::=	<radical> | <component> | <glyph><operator><glyph>
<component> ::=	<radical> | <component><operator><component>
<operator> ::=	- (horizontal) | ^ (vertical) | @ (containing)
<radical> ::=	492 radicals

Mathematically speaking, the representation in Table 1 is productive in that the possible produced outcomes, such as glyphs and components are usually far more than we may accept. Whether the outcome is a legal item or not is up to our choice. Thus, this representation has the property of extensibility that just fit the requirement.

In this productive system, the structure of a glyph is expressed as an expression of radicals. Radical is a basic component which will not be decomposed further. Component is usually referred to the intermediate elements between a glyph and its radicals. An example of glyph structure decomposition is shown in Figure 1. In Figure 1, 臧 and 戕 are components and also glyphs, while 艹, 爿, 戈, and 臣 are radicals, where 臣 is also a glyph.

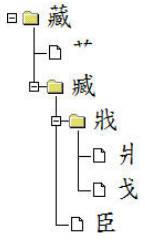


Figure 1. An illustration showing the glyph decomposition of a character "臧".

Example sets of radicals and components are shown in Figure 2 and Figure 3, respectively.

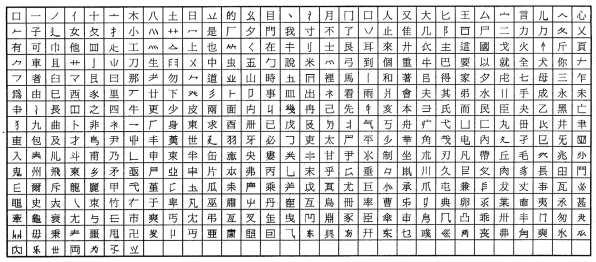


Figure 2. An example set of radicals.

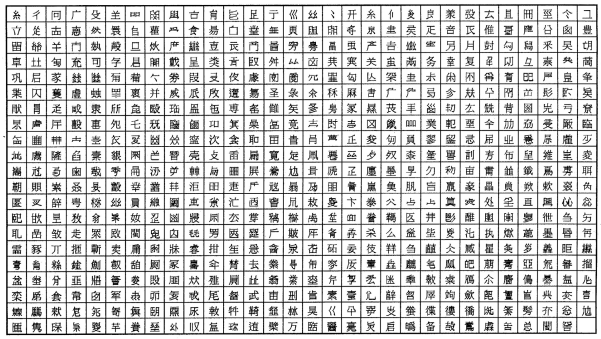


Figure 3. An example set of components.

3.2 Glyph Structure Expression

Glyph structure expression (or *glyph expression* for short) can be used to represent any existing character and those to be created in a user friendly way, which helps people memorize the code sequence very easily. Existing characters including ancient and modern characters can be looked up using the glyph expression. Ancient scripts such as Oracle Bone Inscriptions, Bronze Inscriptions, Seal Inscriptions and their evolutionary relationships can also be represented.

For any new unencoded character to be coined, they can also be created and represented in the same way as existing characters. All one need to know is how to write the glyph in its composing components.

Input methods like Changjie (倉頡) also uses the components to describe a character. In designing an input method, there is a tradeoff between the length of the code sequence representing a character and the number of conflicting characters with exactly the same code sequence. For users, it's important that a code sequence is efficient to type and easy to remember. The number of keystrokes (and thus the length of code sequence) should be minimized. For coding efficiency, on the other hand, the longer the code sequence, the more characters that can be represented, and also the less conflicting characters.

Different from an input method like Changjie, we are not concerned about the number of keystrokes. Instead, the extensibility of characters and the user convenience for looking up characters and creating new ones are the primary consideration. Glyph expressions can represent the most characters with no conflicting representations or ambiguity.

Let us explain how the structure of a glyph is expressed by examples. In Figure 1, the glyph '臧' can be decomposed as:

藏 = 艹 ^ 臧
臧 = 戕 @ 臣
戕 = 爿 – 戈 (1)

In Equation 1, these formulae are generally referred to as *glyph expressions*, in which "^", "-", and "@" represent the operators of vertical, horizontal, and containing compositions, respectively. An expression consisting of only radicals can be obtained by successively decomposing components as follows:

藏 = 艹 ^ 臧
= 艹 ^ (戕 @ 臣)
= 艹 ^ ((爿 – 戈) @ 臣) (2)

In Equation 2, the glyph "藏" is composed of 4 radicals. Note that except 艹, which is encoded in Unicode, the other three radicals are included in Big5 code. Such glyph expression consisting solely of radicals is called a *radical expression*. Similarly, a glyph expression by components such as Equation 1 may be called a *component expression*. When all the operators are eliminated, Equation 2 becomes:

藏 = 艹 爿 戈 臣 (3)

Equation 3 is called the *radical sequence* of the glyph '藏' showing the compositional order of radicals. Following the same naming thought, the formulae in Equation 1 after eliminating operators may be called *component sequences* which show the compositional order of components.

In general, any representation called "expression" refers to complete glyph structure information, while "sequence" does not. Each glyph has a 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 the 9,129 glyphs of [12], there are only 8 pairs of glyphs with exactly the same component sequence, such as (唄, 員) and (峰, 峯). All others can be uniquely identified by their component sequences.

According to our analysis, no character has exactly the same glyph expressions. This shows the uniqueness of glyph expressions with no conflicting characters. Since the representation is natural and intuitive in terms of character writing and composition, people can easily learn to use the glyph expression to represent characters in a short period of time. It's more convenient than user-defined font area since less laborious work needed for users.

3.3 Chinese Glyph Structure Database

Based on the idea of glyph expression model, it is easy to implement the *glyph structure database* (or *glyph database* for short), which contains the knowledge representation of glyph structure information for Chinese characters. For an unencoded character, we can look it up in the glyph database by its glyph expression or from the related characters by component lookup in the user interface of the glyph database as shown in Figure 4. We can easily input an unencoded character in a document. It can also help create characters including ancient ones. With the organization of glyph knowledge, the glyph database is more powerful than a font table.

This database has been extensively expanded and updated since 1993 [7, 11]. The current scale of the character set was enlarged from the set in [12] by including Simplified Chinese characters and the characters in Zhongwen da cidian and Hanyu da zidian[8]. As of the writing of this paper, the glyph structure database contains 59,220 characters of Standard Scripts with 12,681 sets of variants. Other characters include 177 Oracle Bone Inscriptions with 25 variants, 3,459 Bronze Inscriptions with 1,055 variants, 11,100 Small Seal Inscriptions with 1,081 variants, and 372 Chu Bamboo Slips Characters with 73 variants. These ancient characters are precious resources for Chinese studies and research, and more characters are being added.

Containing the rich knowledge of character composition, character evolution, and relational mappings among different forms of characters, the glyph database also provides an integrated user interface for exploring and searching within the database. As shown in Figure 4, both ancient and modern characters can be looked up by components, radicals, or glyph expressions within the glyph database. Then, the components and their structures can be displayed. In addition, the variants, the evolution, and the relations among various forms of the characters can be displayed. The related source information from related lexicons such as Shuowenjiezi can also be looked up.

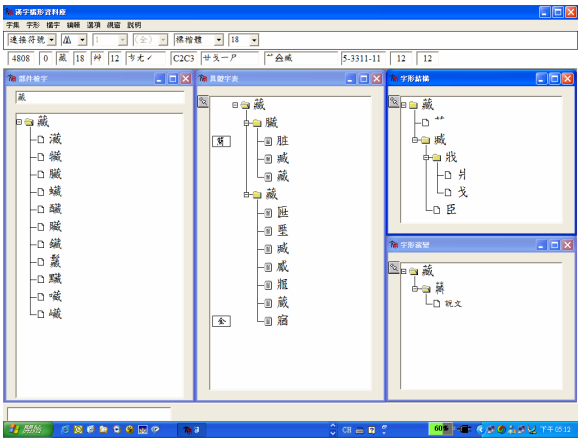


Figure 4. An illustration showing the integrated user interface for the glyph database.

3.3.1 Optimization to the Glyph Database - the Kernel Set

There's a tradeoff between the user-friendliness of glyph expressions and the total number of components in the glyph database. The total number of components in the glyph database has to be minimized for coding space efficiency. On the other hand, to maintain an easy-to-use glyph expression, the average number of components per glyph has to be minimized. Therefore, there's a need for an optimization to the glyph database. This can be done by limiting the operators in a component expression to be one. In this case the number of components per glyph is usually 2, and occasionally 3. 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 database is shown in Figure 5. This kernel

[8] Hanyu da zidian 漢語大字典, Great Dictionary of Chinese Characters, edited by Hanyu da zidian weiyuanhui. Wuhan: Hubei cishu chubanshe and Sichuan cishu chubanshe, 1986..

set includes 457 radicals (in which 269 are also characters and 188 non-character radicals), 629 components, and 1,044 kernel glyphs. There are 2,130 elements in total, where 1,313 are characters and the total frequency of usage is 62.60%.

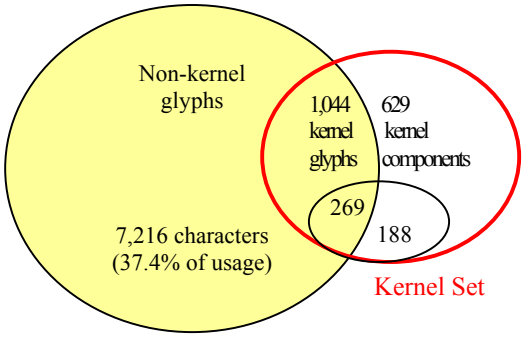


Figure 5. An illustration showing the kernel set of glyph database.

The kernel set in Figure 5 is not very efficient in terms of the total frequency of usage (only 62.60%). This can be further optimized by including frequently used characters into the kernel set. A frequency distribution chart of our glyph database is shown in Figure 6. In Figure 6, it is obvious that including the 1,071 characters of the most frequently used category into the kernel will raise the total frequency of usage of the kernel to 97.8%. The price paid is the increase of the numerical coding space to 3,201. It is not bad and can be afforded by any 2-byte encoding scheme.

Following the same thought, there are 5 possible levels of optimization to the kernel set as listed in Table 2. In Table 2, the fourth choice is recommended which includes all glyphs of the most frequently used and the frequently used categories. It contains 4,946 elements and the average code length is only 1.007 times of a 2-byte code.

Table 2. The weighted average code length at different levels of optimization in the character sets.

Character Set	Number of Codes	The Weighted Average Code Length	Entropy
1. Radicals	457	1.9+1=2.9	7.3038
2. Kernel Set	2,130	(37.4*3+62.6)%=1.748	8.6782
3. Kernel Set & Most Frequently Used Characters	3,201	(2.2*3+97.8)%=1.044	
4. Kernel Set & Frequently Used Characters	4,946	(0.368x*3+99.632)% =1.007	
5. The Whole Set	9,346	1	9.1982

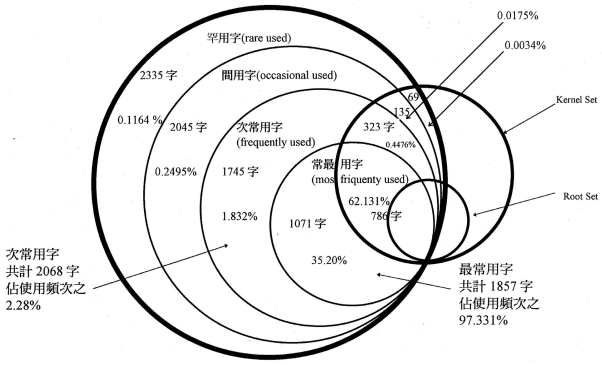


Figure 6. An illustration showing the kernel set and the frequency distribution chart.

3.3.2 A Descriptive Method of Coding

To save the coding space, a descriptive method of coding is adopted for characters not included in the kernel set. In ordinary numeric coding, each character is assigned a numeric code. This works for characters in the kernel set since it is a closed set with a relatively small (2,130) total number of elements. They can be easily encoded into a 2-byte coding space. For characters not included in the kernel set, *descriptive coding* can be used where glyph expressions are used as the code sequences for characters instead of ordinary numeric codes. Descriptive code sequence does not occupy ordinary numerical coding space. Besides, descriptive coding is a productive system which is capable of representing additional newly created characters/glyphs to the existing character set.

Descriptive coding is compatible with existing codes. When 629 components and 188 non-character radicals of the kernel set as shown in Figure 5 are included into an existing encoding scheme such as GB, Big5, or Unicode, the descriptive capability can be provided. 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.

4. Supporting Tools

In order to support the glyph expression and glyph database, tools for character encoding, text input & display, font generation, document searching are required.

A scenario of the lifecycle of processing documents with unencoded characters is illustrated in Figure 7.

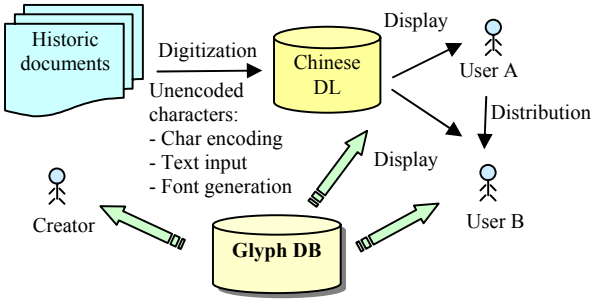


Figure 7. An illustration showing a scenario of the lifecycle of documents containing unencoded Chinese characters.

As shown in Figure 7, the creator of a Chinese digital library has to first digitize the historic documents or antiques such as bronze and stone rubbings. In the digitization process, several issues arise for unencoded characters, for example: character encoding, text input, and font generation, among others. Second, for a user to see the digitized objects, a display tool has to be provided for the unencoded characters to be correctly shown. Third, when User A wants to distribute or disseminate the documents in Chinese digital libraries to User B, the information about the unencoded characters must be carried so that the same characters can be correctly input, displayed, and further utilized. Finally, applications such as document searching can then be supported.

To support the processing of documents containing unencoded Chinese characters with the glyph database, the necessary modules are illustrated in Figure 8.

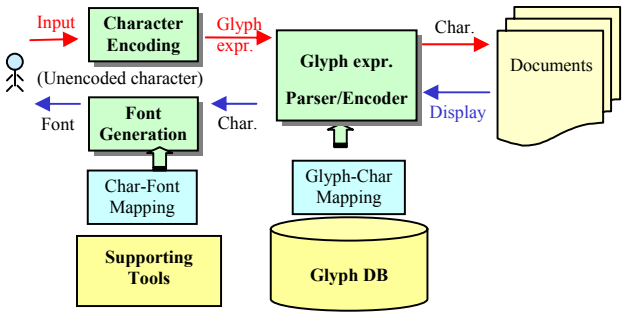


Figure 8. An illustration showing the components in supporting tools for unencoded Chinese characters.

4.1 Tools for Character Encoding and Text Input

As shown in Figure 8, there are two modules involved when inputting texts: *character encoding* and *glyph expression encoder*. In order to represent unencoded characters in ordinary documents, a character encoding module is needed to compose the glyph expression for unencoded characters.This can be obtained either by inputting glyph expressions directly with an unencoded character input method or by relevant glyph lookup with components or radicals from the user interface of glyph database. The user is not required to assign a new code sequence for the unencoded character. As mentioned earlier, since the glyph expression is unique for all glyphs, it can be used as the identifiers for characters.

Then, a glyph expression encoder module is needed to convert glyph expressions into character representation in documents. Markup tags are required to indicate the beginning and ending of the code sequence and glyph expression. Different markup tags might be needed for different kinds of documents such as Web pages and word processing files. Another possible way is to embed font images into the documents if further utilization of unencoded characters is not necessary.

There are several advantages to using glyph structure in encoding scheme. Firstly, the component set as well as the radical set of Chinese characters is a closed system. It will not expand indefinitely as the character set does. Although there are rare cases when they might need to be extended, they are far more manageable than that of character set.

The second advantage is that the glyph structure model is a productive system, as described earlier. This means it has the extensibility and flexibility of not changing the existing system over newly created characters. Thirdly, glyph structure is a kind of knowledge representation for Chinese characters. Characters can be represented by glyph expressions or component sequences. It not only facilitates character encoding and human reading, but also embeds more knowledge about characters into the existing encoding scheme for further applications in Chinese information processing.

4.2 Tools for Font Generation and Display

As shown in Figure 8, two modules are required when displaying texts: *glyph expression parser* and *font generation*. For an unencoded character to be correctly displayed, it has to be first converted from glyph expressions. Then, its associated font has to be generated and the corresponding glyph rendered. A resource locator indicating the location of the corresponding fonts will be encoded in texts.

For a document with glyph expressions, the glyph expression parser is needed to parse the expressions into corresponding characters. For example, after parsing a Web page with markup tags, the glyph expressions are extracted and the corresponding characters are identified. Then, a font generation module is used to get the corresponding fonts for characters in glyph database. For a document with font images, the corresponding font is directly retrieved and rendered. In fact, the glyph database serves as the index for glyph-character mapping and character-font mapping for supporting tools.

4.3 Tools for Document Distribution and Searching

Since the glyph expression is a unique representation of characters, it's natural to distribute the documents in glyph expressions instead of the converted form of glyph fonts or font images. No user-defined fonts are needed. Also, the embedded information about the characters includes the locators of the fonts in glyph database. Information exchange can be done in a more natural and smooth way in which unencoded character problem can be effectively resolved.

As long as the glyph expression is used as the identifier of characters, the unencoded characters in glyph expressions can be searched in the same way as normal characters. Searching of a character can be done by inputting glyph expressions, components, radicals, or relevant glyph lookup with the same input method as in text input.

5. APPLICATIONS IN DIGITAL LIBRARIES AND DISCUSSION

Since 1984, more and more applications in digitizing historic documents and archives had been conducted in Academia Sinica [7, 8, 11]. Currently, the glyph expression and glyph database have been utilized by organizations and projects in various areas including archeology, linguistics, ancient texts, calligraphy and paintings, and stone and bronze rubbings (金石拓片), to name a few.

For example, in Academia Sinica, a full-text database for Chinese ancient documents called Scripta Sinica [19] contains more than 200 million characters of ancient texts such as Ershiwu shi, Shisan jing, … etc. More than 9,727 characters are unencoded [11] if the documents were to be represented in Big5 code. Only 427 of them

cannot be directly represented by our approach. That is, 9,500 (more than 95.6%) of them can be directly resolved without adding new radicals or components.

In NDAP [17], there are currently 9 public institutions participating in the project. These include nearly 100 different databases from libraries, museums, universities, and research institutes in 12 thematic groups such as geology, zoology, botany, anthropology, archives, calligraphy & painting, and rare books. In order to integrate the metadata and the mappings among various databases from such diverse domains, an OAI-based [16] Union Catalog was developed [21]. All services in NDAP can share the same solution in the union catalog. Currently, 140,000 pieces from the Institute of History and Philology of Academia Sinica (IHP)[9] and 10,000 pieces from Han Wooden Slips (漢代簡牘資料庫) are being integrated. The unencoded character problem was among the problems encountered. Glyph database was also utilized to resolve the problem.

In Bronze Inscriptions Research Team (BIRT, 金文工作室) under IHP, a Lexicon of Pre-Qin Bronze Inscriptions of Bamboo Scripts (先秦金文簡牘詞彙資料庫) were constructed where 3,435 characters are unencoded using Big5 code. Among them, 1,462 characters (42.6%) can be directly represented using our approach. The remaining 1,973 unencoded characters together with 2,253 more characters collected from other references on Bronze Inscriptions were added into the glyph database.

As of this writing, the December 2004 version of the glyph database was announced and included in a CD-ROM. The database, tools, documents, papers, and related information are also available for download in the following website: http://www.sinica.edu.tw/~cdp/.

Results

As an example result, the original message of a short passage (in Bronze Inscriptions) scripted on a Bronze ritual vessel is shown in Figure 9.

(a)

(b)

“白（伯） 父乍（作）／寶鼎，其子子／孫孫永用。〔幷〕。”

(c)

“白（伯）覲父乍（作）／寶鼎，其子子／孫孫永用。〔幷〕。”

Figure 9. An illustration showing the short passage scripted on a Bronze ritual vessel (伯覲父鼎): (a) the original image, (b) the message displayed in Big5 code, and (c) the message in glyph expression.

As shown in Figure 9, the unencoded character “覲“cannot be displayed in Big5, while it can be correctly displayed in glyph expression. More examples such as Mao-Kung Ting (毛公鼎) are available upon request.

Discussion

User-defined font area or private user area that most existing systems adopt is only a small-scale short-term solution to the unencoded character problem. It’s neither scalable, nor portable.

Our approach identifies each character with its glyph expression, which is extensible to large-scale full-text Chinese databases or digital library applications. Also, the font is stored as a centralized resource on the Web, and only the resource locators are required to be embedded in documents. This approach helps resolve the portability issue.

The glyph expression model is extensible in that a finite set of components can be used to derive an open set of characters, including all possible forms from the past and those to be coined in the future. The current glyph database is built on top of existing Big5 coding space. The integration with other coding schemes such as Unicode is in progress.

6. CONCLUSION

In this paper, we proposed a glyph expression model for unencoded Chinese characters in digital libraries. The glyph expression model and the glyph database were introduced to resolve the unencdoed character problem. A scenario of dealing with documents containing unenecoded character was illustrated as well as the supporting tools for applying the model in character encoding, font generation, text display and input, and document dissemination. The statistics on current applications in Academia Sinica, NDAP program, and research teams on various domains also showed its potential in more applications in Chinese digital libraries.

7. ACKNOWLEDGEMENTS

This work was supported in part by the following grants: NSC 93-2752-E-001-001-PAE, 93-2422-H-001-0004 and 93-2213-E-001-025. We would like to express our heartily thanks to many researchers and scholars, such as Xue, Yong-Cheng (許永成), Jung, Bor-Sheng (鍾柏生) from IHP, Academia Sinica, Chi, Shiu-Sheng (季旭昇) from Department of Chinese, National Taiwan Normal University, and Wang, Ning (王寧) from Beijing Normal University. We have been cooperating with them for years for sharing the knowledge of Chinese Etymology (or Wen-Zi Xue, 文字學). The study and the system would not be possible without the efforts and supports from them.

8. REFERENCES

[1] Bishop, T. and Cook, R.S. A Specification for CDL Character Description Language. *In Glyph and Typesetting Workshop*, Kyoto, Japan, 2003.

[2] Cook, R.S. The Extreme of Typographic Complexity: Character Set Issues Relating to Computerization of the Eastern Han Chinese Lexicon Shuowenjiezi. In *Proc. of the 18th International Unicode Conference (IUC-18)*, Apr. 2001.

[9] IHP (the Institute of History and Philology of Academia Sinica), http://www.ihp.sinica.edu.tw/.

[3] Cook, R.S. Typological Encoding of Chinese: Characters, Not Glyphs. In *Proc. of 19th International Unicode Conference (IUC-19)*, Sep. 2001.

[4] Ho, C. W. CHANT (CHinese ANcient Texts): a Comprehensive Database of All Ancient Chinese Texts up to 600 AD, *Journal of Digital Information, Volume 3 Issue 2*, Article No. 119, Aug. 2002.

[5] Hsieh, Ching-Chun. The Missing Character Problem in Electronic Ancient Texts (電子古籍中的缺字問題). In *the First Conference on Chinese Etymology*, Tianjin, Aug. 25-30, 1996. (in Chinese) http://www.sinica.edu.tw/~cdp/paper/1996/19960825_1.htm

[6] Hsieh, Ching-Chun. The Glyph and Encoding in Hanzi – On Redesigning Hanzi Interchange Code -- Part 1 (漢字的字形與編碼). In *International Conference on Hanzi Character Code and Database*, Kyoto, Oct. 4, 1996. (in Chinese) http://www.sinica.edu.tw/~cdp/paper/1996/19961004_1.htm

[7] Hsieh, Ching-Chun (謝清俊). A Descriptive Method for Re-engineering Hanzi Information Interchange Codes – On Redesigning Hanzi Interchange Code -- Part 2. In *International Conference on Hanzi Character Code and Database*, Kyoto, Oct. 1996. http://www.sinica.edu.tw/~cdp/paper/1996/19961005_1.htm.

[8] Hsieh, Ching-Chun and Lin, Shih. A Survey of Full-text Data Bases and Related Techniques for Chinese Ancient Documents in Academia Sinica (中央研究院古籍全文資料庫的發展概要), *International Journal of Computational Linguistics and Chinese Language Processing, Vol. 2, No. 1*, Feb. 1997. (in Chinese) http://rocling.iis.sinica.edu.tw/CLCLP/Vol2-1/a5.htm

[9] Hsieh, Ching-Chun. On the Formalization and Search of Glyphs in Chinese Ancient Texts (談古籍檢索的字形問題). In *Conference on Rare Book and Information Technology*, Taipei, Apr. 21, 1997. (in Chinese) http://www.sinica.edu.tw/~cdp/paper/1997/19970421_1.htm

[10] Jenkins, J.H. The Dao of Unihan. In *Proc. of the 26th International Unicode Conference (IUC-26)*, Sep. 2004.

[11] Juang, Derming, Hsieh, Ching-Chun, and Lin, Shih. On Resolving the Missing Character Problem for Full-text Database for Chinese Ancient Texts in Academia Sinica (中央研究院古籍全文資料庫解決缺字問題的方法). In *the Second Cross-Strait Symposium on the Rectification of Ancient Texts*, Beijing, May 11-13, 1998. (in Chinese) http://www.sinica.edu.tw/~cdp/paper/1998/19980511_1.htm

[12] Lin, S. (林樹). Research on the Fundamental Chinese Character Set for Computer Use (中文電腦基本用字研究), Technical Report, Department of Computer and Control Engineering, NCTU, March 1972. (in Chinese)

[13] Liu, W. The Development of Digital Collections and Metadata Applications in Chinese Libraries. In *Proc. of International Symposium on Digital Libraries and Knowledge Communities in Networked Information Society (DLKC 2004)*, Japan, Mar. 2004.

[14] Lu, Q. (陸勤). The Ideographic Composition Scheme and Its Applications in Chinese Text Processing. In *Proc. of the 18th International Unicode Conference (IUC-18)*, Apr. 2001.

[15] Lu, Q., Chan, S., Li, Y., and Li, N. Decomposition for ISO/IEC 10646 Ideographic Characters. In *the 3rd Workshop on Asian Language Resources and International Standardization, COLING 2002*, Taipei, 2002.

[16] OAI (Open Archives Initiative), http://www.openarchives.org/.

[17] NDAP, National Digital Archives Program, Academia Sinica (http://www.ndap.org.tw/)

[18] Ni, K (倪耿). 中國文字之結構模式及其分析, Master's Thesis, Institute of Electronics, NCTU, 1972. (in Chinese)

[19] Scripta Sinica, Hanji dianzi wenxian 漢籍電子文獻, Academia Sinica, http://www.sinica.edu.tw/~tdbproj/handy1/.

[20] The Unicode Consortium. The Unicode Standard, Version 4.0.1, defined by: *The Unicode Standard, Version 4.0* (Reading, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1), as amended by Unicode 4.0.1.

[21] Union Catalog of NDAP, http://catalog.ndap.org.tw/.

[22] Wittern, C. Chinese Buddhist texts for the new Millennium — The Chinese Buddhist Electronic Text Association (CBETA) and its Digital Tripitaka. *Journal of Digital Information, Volume 3, Issue 2*, Article No. 123, Sep. 2002.

[23] Wittern, C. and App, U. IRIZ Kanji Base: A New Strategy for Dealing with Missing Chinese Characters. In *EBTI (The Electronic Buddhist Text Initiative)*, Taipei, April 1996.

[24] Yang, G. and Zhang, T. The Development of the China Digital Library. *Electronic Journal of Academic and Special Librarianship, Vol. 3, No. 3*, 2002.

An empirical study of Chinese word-word language directed network

Jianwei Wang, Lili Rong
Institute of Systems Engineering
Dalian University of Technology
Dalian, China
E-mail: wdut@yahoo.cn, llrong@dlut.edu.cn

Tao Jin
School of Electronic and Information Engineering
Dalian University of Technology
Dalian, China
E-mail: dlutjintao@hotmail.com

***Abstract*—In this paper, we explore the statistical properties of Chinese language network within the framework of complex network theory. Based on one of the common databases in China, i.e. Tshinghua Unisplendour's one, we construct Chinese word-word language directed network (CWWLDN). The node in CWWLDN represents a single character, and while a directed link between two nodes exists if two different characters appear in the same words. The empirical results show that CWWLDN possesses scale-free behavior and small-world effect. Two characteristics indicate that CWWLDN is similar to other previously studied language networks, which shows that different languages may have some common properties in their evolutionary processes. We believe that our research may shed some new light into the Chinese language evolution and find some potentially significant implications.**

Keywords-component; complex network; scale free; small world; direct network

I. INTRODUCTION

Over the last few years, it has been suggested that a lot of social [1,2], technological [3], biological [4], and information [5,6] networks share the following three striking statistical characteristics: scale-free, high clustering coefficient and small average path length (APL). Scale-free shows that the degrees of nodes on the networks satisfy a power law distribution: $P(k) \sim k^{-\gamma}$, where k is the number of links of a randomly chosen node in the network and γ is the scaling exponent. High clustering coefficient indicates that if vertices A and B are linked to vertex C, then A and B are also likely to be linked to each other. Small APL implies that the expected number of edges needed to pass from one arbitrarily selected node to another one is low, that is, APL grows logarithmically with the number of nodes or slower.

In the past few years, the phenomenon of human language is widely studied from various points of view [7-10]. Moreover, with the expansion of complex network research, it is interesting not only for social scientists, anthropologists or philosophers, but also for those, interested in researching on complex networks. Any language is composed of many thousands of words linked together in an apparently fairly sophisticated way. Because words are a good example of simple elements that are combined to form complex structures, such as novels, poems, dictionaries, and manuals that are designed to transport or convey information, human languages have been much investigated from the perspective of complex network. Identifying and understanding the common network topology of languages is of great importance, not only for the study of languages themselves, but also for cognitive science where one of the most fundamental issues concerns associative memory, which is intimately related to the network topology. However, on the one hand, there were very few works about the topology study on the Chinese language, although it is one of the most widely used languages in the world. On the other hand, constructing the networks, the methods were often based on two ways [11-13]: (i) a node represents a word; (ii) a link exists between two words if they express similar concepts or appear simultaneously in a sentence. While the researches on a word as a node of networks and the relation between a word and a word are ignored usually. In fact, to the extent, the people's way of thinking is well discussed by this method.

In this paper, we define a single Chinese character as a word, such as "中", "国", and "人"; while the words which consists of two or more Chinese characters, such as "中国", "人民", and "网络". If we consider word-word relation as our method constructed network and treat this network as a finite directed network in which a single word is a node and two nodes are linked if they are neighbors, then we can analyze this network completely. Inspired these, Chinese word-word language directed network (CWWLDN) is proposed, which is based on the database of Tsinghua Unisplendour, one of the most common databases. We present a comprehensive analysis of the network characteristics including degree distribution, clustering coefficient, and average path length. Our results demonstrate that the degree distribution of CWWLDN follows a power-law distribution, which shows that it exhibits the scale-free property. And the small APL and high clustering coefficient indicates that it possesses the small-world effect. Our study makes it possible to investigate the complexity of the Chinese language within the framework of network theory.

This work was supported by the National Natural Science Foundation of China under Grant nos. 70571011 and 70771016.

978-1-4244-2013-1/08/$25.00 ©2008 IEEE

The rest of this paper is organized as follows: in section 2, we introduce the data source and present network construction method in detail. The statistical characteristics and results are proposed in Section 3. Finally, some summaries and conclusions are shown in Section 4.

II. CONSTRUCTION OF CHINESE WORD-WORD LANGUAGE DIRECTED NETWORK

For better understanding the formation of Chinese language and analyzing various topology characteristics of it, it is necessary that we start with seeking for a typical database, and then construct Chinese word-word language directed network based on it.

A. *The Database*

Chinese is one of the most widely used languages in the world. Chinese characters play an important role in its well-known civilization. One reason is that they bear some unique and elegant structures. Most western language characters are phonetic-based, while Chinese characters are mostly picture-based. In order to discuss characteristics of the structure of Chinese characters, in this paper, we make use of the database from Tsinghua Unisplendour, one of the most common databases. The chosen database is representative, since it includes 548,387 words, covering almost all words.

Among the 548,387 words, many words covered by three or more Chinese characters, have a certain degree of data redundancy, because of taking into account the relations between a word and a word, for example, "复杂网络", no relation between "杂" and "网". So we will tackle with the data in the database from Tsinghua Unisplendour in order to keep those words consisting of two Chinese characters.

B. *Network Construction Method*

According to the principle to extracting words consisting of two characters, the Chinese characters studied in this paper contain 7440 characters and 178,971 words from Tsinghua Unisplendour' database.

The CWWLDN is established as follows:

(i) We define every one among the 7440 characters as a node;

(ii) Two nodes are connected if they appear in the same words.

Thus, the CWWLDN may shed some insight into the understanding of the structure characteristics of words formation in Chinese language. In Fig.1, we present a simple example of 45 words consisting of two Chinese characters and construct a small CWWLDN based on Fig.1.

心脏 中心 中国 国人 人民 民众 发展 展开 开心
发现 展现 开发 凤中 民风 高中 关中 关心 开关
国中 华中 口中 众口 人口 心口 开口 风口 中大
大小 中发 中方 中风 地方 大地 地界 土地 民心
民国 小人 大人 高人 高地 人心 中展 高开 民口

Figure 1. 45 words consisting of two Chinese characters.

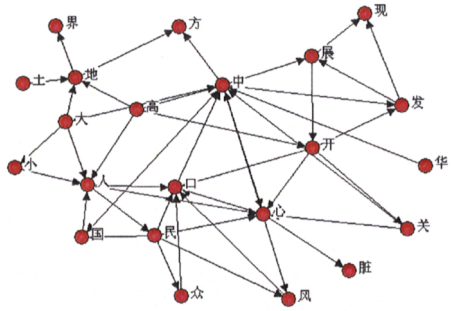


Figure 2. An example of the CWWLDN based on the 45 words shown in Fig.1.

Following the constructional method, we get the CWWLDN, which includes 7440 nodes and 178,971 edges, and contains one giant component of 7125 nodes and 178,770 edges. We also delete the ones unconnected with other words from the CWWLDN, among which including the words constructed by the same two Chinese characters, the number of which is only 87, such as "座座", "尊尊", and "醉醉". In the following, in order to investigate the structure characteristics of the CWWLDN, we will only consider the topology properties of the giant connected component of the CWWLDN ignoring the unconnected units.

III. EMPIRICAL RESULTS

We focus on the topological characteristics of the CWWLDN, including degree distribution $P(k)$, APL, and clustering coefficient.

A. *Degree Distribution*

The degree distribution is one of the most important statistical characteristics of a network. By definition, the degree k_i of a node i is the number of edges incident from i, and is defined in terms of the adjacency matrix A as:

$$k_i = \sum_{j \in N} a_{ij}$$

If the graph is directed, the degree of the node has two components: the number of outgoing links $k_i^{out}=\Sigma_j a_{ij}$ (referred to as the out-degree of the node), and the number of ingoing links $k_i^{in}=\Sigma_j a_{ji}$ (referred to as the in-degree of the node). The total degree is then defined as $k_i=k_i^{out}+k_i^{in}$.

Degree distribution $P(k)$ is the probability that a randomly selected node has exactly k edges. For many real complex networks, $P(k)$ follows a power-law distribution: $P(k)\sim k^{-\gamma}$. The networks with a power-law distribution are called scale free.

In order to understand the topology characteristics of Chinese word-word language directed networks, we investigate two distributions, $P(k^{in})$ and $P(k^{out})$ (see Fig.3 and Fig.4).

Identify applicable sponsor/s here. If no sponsors, delete this text box. (sponsors)

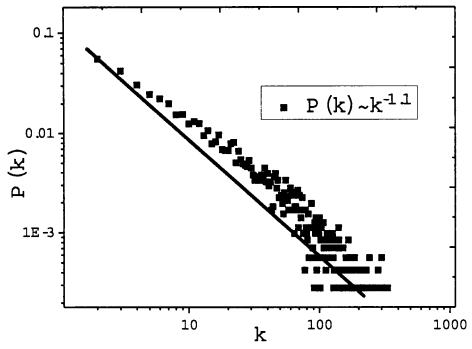


Figure 3. In-degree distribution of the CWWLDN.

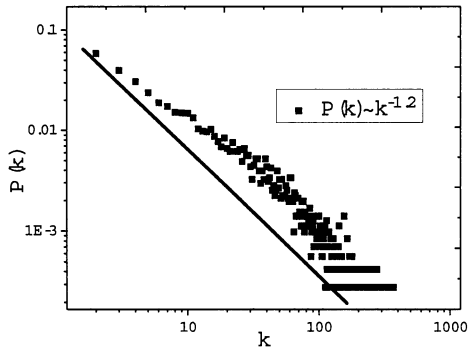


Figure 4. Out-degree distribution of the CWWLDN.

In Fig.3 and Fig.4, we report the in-degree and the out-degree distributions of the CWWLDN, which includes 7125 nodes and 178,770 edges. We get the values of the parameter γ of two distributions respectively, which follows a power-law distribution: $P(k)\sim k^{-\gamma}$, based on the slope of the fitting lines, value γ is about -1.1 in the in-degree distribution and about -1.2 in the out-degree one. In Fig.5, we further discuss the total degree distribution of the CWWLDN (see Fig.5), where value γ is about -1.15.

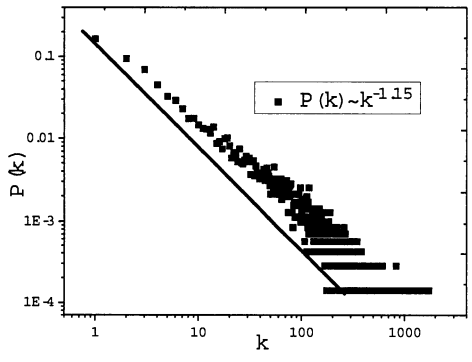


Figure 5. Degree distribution of the CWWLDN.

Based on the empirical results, the considered CWWLDN obviously exhibits a scale-free behavior. Moreover, one can easily see that the degree spectrum is continuous and has a fat tail for large degree values, in agreement with the analytical results to a relatively homogeneous topology similar to most scale-free networks.

As a matter of fact, because of language characteristic and convenient communication, some characters used frequently become the hubs that connect other characters. Frequency of occurrence of a character leads to the emergency of scale-free behavior and the phenomenon of "the rich get richer" of the CWWLDN.

B. Clustering Coefficient

Clustering [14], also known as transitivity, is a typical property of acquaintance networks, where two individuals with a common friend are likely to know each other. Most real-life networks show a cluster structure which can be quantified by the clustering coefficient. The clustering coefficient of a node gives the relation of connections of the neighborhood nodes connected to it. By definition, clustering coefficient C_i of a node i is the ratio of the total number e_i of existing edges between all k_i its nearest neighbors and the number $k_i(k_i-1)/2$ of all possible edges between them, i.e. $C_i=2e_i/k_i(k_i-1)$. The clustering coefficient C of the whole network is the average of all individual C_i's:

$$C =< c >= \frac{1}{N}\sum_{i\in N} C_i .$$

By definition, $0\le C_i\le 1$, and $0\le C\le 1$.

We calculate the clustering coefficient of the CWWLDN and find that the networks have a relatively high clustering coefficient. Compared with the clustering coefficient $C_{rand}=<k>/N=0.003521$ in agreement with the same scale random network, the clustering coefficient $C=0.245169$ of the CWWLDN is about 70 times higher than that the random network.

The high clustering coefficient may be explained by the people who would like to use some familiar Chinese characters to create new words. Thus, the characters frequently used are connected to each other, such as "中" and "心", while the characters less used do not tend to form the clustering, such as "铿" and "锵", no connection with other characters, respectively. From the characteristic of the CWWLDN, we can see that the elements of the new words appear in the common characters, which will lead to the emergency of more clustering nodes.

C. Average Path Length

Shortest paths play an important role both in the transport and communication within a network and in the characterization of the internal structure of the network, which is defined as the length of the geodesic from one node to the other. Average path length (APL) is defined based on the concept of shortest path, which is the average value of shortest path length of all node pairs in the networks. APL is one of the most important properties in measuring the efficiency of

communication networks. In a store-forward computer network, for example, the most useful measure characterizing the performance is the transmission delay in sending a message through the network from the source to the destination. The transmission delay is approximately proportional to the number of edges that a message must pass through. Therefore, APL plays a significant role in measuring the transmission delay. Moreover, it is well known that the most important property of a small-world network is a logarithmic APL with the number of nodes. Hence, its study has attracted much attention.

Since the original conception of small-world effect is defined based on undirected networks, hereinafter we only consider the undirected version of CWWLDN. If we use the value of d_{ij} to represent shortest path length from node i to node j, mathematical expression of APL, also known as characteristic path length, is defined as [15]:

$$L = \frac{1}{N(N-1)} \sum_{i,j \in N, i \neq j} d_{ij},$$

where N represents the total number of all the nodes in the networks.

We get the value APL of the CWWLDN in the case of ignoring the direction of edges, which is about 2.73. Compared with the number $N=7125$ of nodes in the CWWLDN, 2.73 is very small. It is obvious that random chosen two characters from the CWWLDN exist in the correlations from each other by a few steps. The reason why the APL is so small may be related to the existence of hubs, which are bridges between different nodes of the network that would otherwise be separated by many links.

In the last few years, a series of empirical studies report that many real complex networks possess small-world property including biological and technological ones, which is the unifying concept constituting our basic understanding of the organization of real-life complex systems.

Small-world effect is measured by clustering coefficient and APL. The CWWDLN is considered as the small-world network, since which possesses high clustering coefficient and small APL. The emergency of this phenomenon can be better explained by the evolution of Chinese language network. As the network grows, since the existing some nodes that have more connections have opportunity to attach to the new node according to human's cognitive principle, these nodes become the "bridge" ones naturally, i.e., the "shortcuts" ones, which drastically reduces the APL, leading to a small-world behavior.

IV. CONCLUSION

We have presented a comprehensive investigation on the statistical properties of the CWWLDN within the framework of complex network theory. Like other many complex networks from nature and society, the CWWLDN also possesses to show similar statistical features: scale-free behavior and small-world effect. We try to explain the empirical results from linguistics and the method to using characters, which may be useful to study the dynamics of Chinese language networks.

Although the results reported in this paper represent only the starting point towards the understanding of Chinese language networks, it could be relevant for a more realistic modeling of the Chinese language networks, and could find other implications.

Further studies planned to elucidate remaining issues which can also be seen as pointing to the potential problems and limitations of the current approach, are:

(1) There exist other methods to the construction of Chinese language networks, such as a syntactic network;

(2) We may further discuss the dynamics of the words' evolution and the effect of weights of edges in the CWWLDN. In summary, for better understanding of the language science, it is necessary that we explore in depth the correlations among the characters.

ACKNOWLEDGMENT

Jianwei Wang would like to thank the database provided by Tsinghua Unisplendour.

REFERENCES

[1] M.E.J. Newman, "Scientific collaboration networks: II. Shortest paths, weighted networks, and centrality", Phys. Rev. E 64, 2001, 016132.

[2] Newman M E J, "Scientific collaboration networks. I. Network construction and fundamental results", Phys. Rev. E 64, 2001, 016131.

[3] Xu T, Chen R, He Y, et al., "Complex network properties of Chinese power grid", International Journal of Modern Physics B 18, 2004, pp. 2599-2603.

[4] Jeong H, Mason S, Barabási A-L. et al., "The largescale organization of metabolic networks", Nature 407, 2000, pp. 651-654.

[5] Albert R, Jeong H, Barabási A-L, "Diameter of the World Wide Web", Nature 401, 1999, pp. 130-131.

[6] Vázquez A, Pastor-Satorras R, Vespignani A, "Large-scale topological and dynamical properties of the Internet", Phys. Rev. E 65, 2002, 066130.

[7] R.F. Cancho, R. Solé, "The small world of human language", Proc. R. Soc. London B 268, 2001, pp. 2261-2265.

[8] A. E. Motter, P. S. de Moura, Lai Ying-Cheng, P. Dasgupta, "Topology of the conceptual network of language", Phys. Rev. E 65, 2002, 065102.

[9] M. Steyvers, J. B. Tenenbaum, "The large scale structure of semantic networks: statistical analyses and a model of semantic growth", Cogn. Sci. 29, 2005, 41.

[10] M. A. Nowak, D. C. Krakauer, "The evolution of language", Proc. Natl. Acad. Sci. USA 96, 1999, 8028.

[11] Jianyu Li, Jie Zhou, "Chinese character structure analysis based on complex networks", Physica A 380, 2007, pp. 629-638.

[12] Shuigeng Zhou, Guobiao Hu, Zhongzhi Zhang, Jihong Guan, "An empirical study of Chinese language networks", Physica A 387(2008) 3039-3047.

[13] Haitao Liu, "The complexity of Chinese syntactic dependency networks", Physica A 387, 2008, pp. 3048-3058.

[14] M.E.J. Newman, "The structure of scientific collaboration networks," Proc. Natl. Acad. Sci. USA 98, 2001, pp. 404-409.

[15] Watts D J, Strogatz S H, "Collective dynamics of 'small-world networks", Nature 393 (6684), 1998, pp. 440-442.

2014 11th IAPR International Workshop on Document Analysis Systems

Graph Model Optimization based Historical Chinese Character Segmentation Method

Jingning Ji **Liangrui Peng** **Bohan Li**
Tsinghua National Laboratory for Information Science and Technology
Dept. of Electronic Engineering, Tsinghua University, Beijing, China
zjwljjn@gmail.com **plr@ocrserv.ee.tsinghua.edu.cn** **jim.jd.davis@gmail.com**

***Abstract*—**Historical Chinese document recognition technology is important for digital library. However, historical Chinese character segmentation remains a difficult problem due to the complex structure of Chinese characters and various writing styles. This paper presents a novel method for historical Chinese character segmentation based on graph model. After a preliminary over-segmentation stage, the system applies a merging process. The candidate segmentation positions are denoted by the nodes of a graph, and the merging process is regarded as selecting an optimal path of the graph. The weight of edge in the graph is calculated by the cost function which considers geometric features and recognition confidence. Experimental results show that the proposed method is effective with a detection rate of 94.6% and an accuracy rate of 96.1% on a test set of practical historical Chinese document samples.

Keywords—historical Chinese document; character segmentation; graph model

I. INTRODUCTION

Digitization of historical Chinese documents, e.g. Dunhuang manuscripts, is important for preservation, transmission and research of Chinese history. Transcribed during the 4th century to the 11th century, the Dunhuang manuscripts include more than 60,000 titles. Some of the historical documents are unique, which is of great significance to the complementary of Chinese history and Buddhist culture. With the effort of International Dunhuang Project (IDP), Dunhuang document images are shared via websites. However, most of these document images cannot be searched in full text due to the lack of recognition method of Chinese historical documents.

Character segmentation is an essential process in Chinese historical document recognition. However, the complex character structure, broken or touching characters, noise disturbance and different layouts in historical Chinese documents make it difficult to segment characters correctly. Fig. 1 shows an example of the Dunhuang documents which have the above-mentioned challenges in character segmentation.

Researchers have proposed a variety of character segmentation methods. As is discussed in [2], segmenting strategies can be categorized to "dissection", recognition-based methods, holistic methods and combinations of these three. The "dissection" method usually utilizes geometric features to find character segmentation results, which can be applied to historical Chinese character segmentation. However, purely recognition-based or holistic methods are not suitable for historical Chinese characters, as historical Chinese characters have a large character set up to tens of thousands of characters.

Most Chinese character segmentation algorithms include

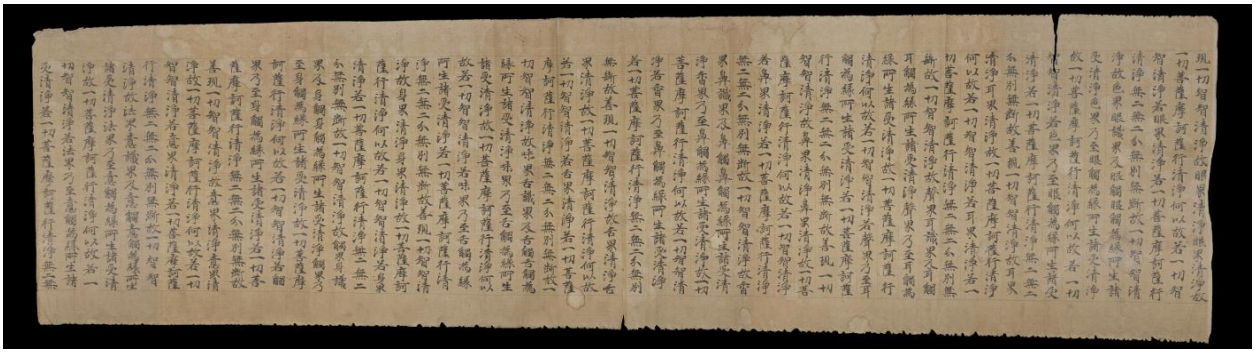


Fig. 1. An example of the Dunhuang documents transcribed during the 4th century to the 10th century.

978-1-4799-3243-6/14 $31.00 © 2014 IEEE
DOI 10.1109/DAS.2014.57

over-segmentation and merging stages. In over-segmentation stage, projection profiles, connected components, Viterbi algorithm [3, 4], Voronoi diagram [5] and stroke analysis [7, 8, 9] are mainly used.

In merging stage, a lot of algorithms have been proposed. Yi-Hong Tseng et al. [3] used the shortest path finding algorithm. Van Phan [5] proposed a criterion for removing Voronoi edges. Xiaolu Sun et al. [8] applied the Viterbi algorithm to achieve local optimization. Shuyan Zhao, et al. [9] extracted 15 geometric features for the decision tree generation to classify merging path. Several fuzzy decision rules were introduced as well to improve the performance. In the work of Lin Yu Tseng et al. [7], four kinds of merging operations are used to merge the stroke bounding boxes into candidate boxes. Then the candidate boxes are merged into characters by a dynamic programming method.

In this paper, a novel method for the merging process is proposed. The candidate segmentation positions are denoted by the nodes of a graph, and the merging process is regarded as selecting an optimal path of the graph. The weight of edge of the graph is calculated by the cost function considering component height, component gap, maximum component length, aspect ratio and recognition confidence. Different path selecting strategies is tested on practical Dunhuang historical Chinese document samples provided by National Library of China.

The rest of this paper is organized as follows: Section II presents our proposed methodology, including graph model of over-segmentation results, cost calculation and path selecting strategies. Experimental results and error analysis are discussed in Section III and conclusion is in Section IV.

II. METHOD

The overall OCR system is based on a prototype of machine-printed Chinese recognition system. The input RGB historical Chinese document image is binarized first. After binarization and noise removal, the system analyzes the layout of the image and segments the image into different regions. The regions are then segmented into text-line images. The text-line image is segmented into small components based on projection, connected component analysis and touching stroke analysis [8]. As these small components are over-segmented, a merging process is needed to find optimal segmentation results. Finally the merged characters are sent to a Chinese character recognizer to be recognized. Fig. 2 shows the process of the system.

The performance of the merging process is crucial to the overall performance of whole system. To solve the merging problem, the principle of graph theory is introduced as the following.

A. Graph model of over-segmentation results

The over-segmentation results of the text-line image can be treated as a directed graph where its segmentation positions can be treated as nodes and its small components can be treated as edges. The direction of edge is from the upper segmentation positions to the lower segmentation positions

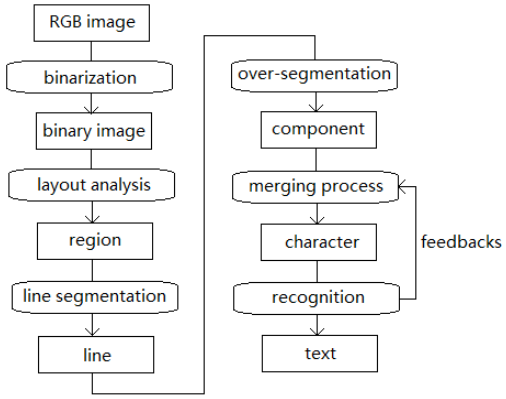
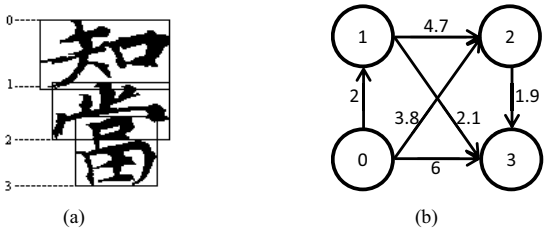


Fig. 2. The system flowchart.



(a) (b)

Fig. 3. Transform from a line image to a graph: (a) A typical line image. It is over-segmented to 3 small images, corresponding to 4 cuts including start and end. (b) The graph transformed by line image.

and the weight of edge is determined by component features. We call that the cost of a component, which describes the degree of difference between the component and a normal character. The cost should be non-negative in accordance with its meaning and for the convenience of application of graph theory algorithms. A typical transform from a part of a text-line image to a graph is shown in Fig. 3.

B. Cost calculation

Component height, component gap, maximum component length, aspect ratio and recognition feedback are considered to sum up the cost. The followings are specific costing methods.

1) Component height

The average height of all components is calculated first. Then the height of each component is divided by the average height. The normalized height h should approach to a certain value if the component is a real character. A function f(h) is designed to calculate the cost of height as shown in Fig. 4.

$$f(h)=\begin{cases}\alpha_1(h-th_1)^2 & (\mathrm{h}\le th_1)\\ 0 & (th_1<h<th_2)\\ \alpha_2(th_2-h)^2 & (\mathrm{h}\le th_2)\end{cases} \quad (1)$$

where th_1 and th_2 are lower and upper thresholds of a normal character height. α_1 and α_2 are used to adjust the proportion of cost generated by component height. The quadratic function is chosen indicating the mean square error.

For the convenience of analyzing, we set th_1=th_2=TH. As the value of TH is important for our method, it is discussed in Section III.

2) Component gap

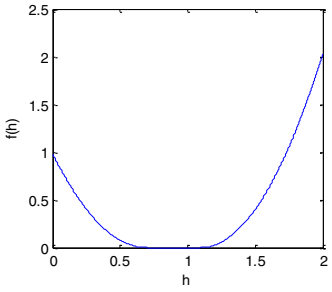


Fig. 4. An example of f(h) when th_1=0.7, th_2=1.1, α_1=2 and α_2=2.5.

It's obvious that a wider gap between components means greater possibility of a cut, indicating smaller costs of both components. As the cost should also be non-negative, the cost function of character gap S can be defined as

$$g(S_1, S_2) = \beta \times e^{-S_1/s} + \beta \times e^{-S_2/s} \quad (2)$$

where S_1 and S_2 are gaps between current component and its former and latter component, s the size coefficient and β the proportional coefficient. The parameter s relies on the character size and can be determined by the average height of all components. The parameter β relies on the intensity of characters.

It should be noticed that the gap should not be the space between bounding boxes, but the shortest distance between two black connected components.

3) Maximum component length

The maximum component length here refers to the greater value of component height and width, i.e. L=max(H,W) where H is image height and W is image width. It is used to add cost of small noises or dot strokes which have similar height to some special characters such as "—". The cost function of maximum component length can also be expressed by f(h) with a wider threshold and a smaller proportion. The parameter is decided empirically in our experiment.

4) Aspect ratio

The aspect ratio of a component is useful information when characters have different sizes but similar shapes. It is simply defined as the ratio of width to height and its cost function is similar to f(h).

5) Character recognition

Feedbacks of character recognition are of great help to describe the similarity between the component and a character. Our Chinese character recognizer is a MQDF classifier. As is discussed in [1], the following equation is an ideal estimation of recognition confidence.

$$e = 1 - \frac{d_i}{\min_{k \neq i}(d_k)} \quad (3)$$

$$d_j = \min_{1 \leq k \neq K_j} \left\| y - y_k^j \right\| \quad (4)$$

where e is the confidence, i the classification result, K_j the number of prototypes in class j, y the features of image, y_k^j the features of prototype k in class j. The cost function can be

$$h(e) = \gamma(1-e) = \gamma \times \frac{d_i}{\min_{k \neq i}(d_k)} \quad (5)$$

where γ is the proportional coefficient depending on the performance of recognizer. If the recognition result is good enough, γ should be very large to reach the optimum.

The introduction of recognizer helps the system correctly merge most of characters which can be recognized, thus promoting the final recognition rate. In addition, recognizer can detect special characters such as "—" whose threshold in cost function should be adjusted to calculate a better cost.

C. Path selection

The merging process is equivalent to finding a proper path in the established graph model. The path starts from s_0 and ends at s_n and goes through edges which correspond to the merged components. If the costs of components are well calculated, the path with the minimum total cost is the optimal result. The shortest path and the minimum average cost path are considered.

Lots of shortest path algorithms have been proposed in graph theory and the Dijkstra algorithm is adopted in our method. The steps of Dijkstra algorithm are as follows.

Step 1. Let S denote the set of visited nodes and initialize S={s_0} where s_0 is the beginning node. Let T denote the set of unvisited nodes. Save the tentative distance from each node s_k to s_0 according to the equation $d(s_k)=C(s_0,s_k)$ where $C(s_0,s_k)$ means the weight of edge between s_0 and s_k, and $d(s_k)$ the tentative distance from s_0 to s_k. Set the beginning node s_0 as current node.

Step 2. For the current node s_c, calculate the tentative distance of its unvisited neighbor nodes using $d(s_k)=d(s_c)+C(s_c,s_k)$. Update $d(s_k)$ if it is less than the previously recorded tentative distance. Take the current node s_c from T to S. Choose node in T with minimum tentative distance as the current node.

Step 3. Repeat step 2 until the destination node s_n is visited. The current tentative distance $d(s_n)$ is the value of shortest path.

Sometimes the shortest path has its problem that multiple true characters have greater total cost than their merged component. The minimum average cost algorithm is proposed to overcome the disadvantage. Its process is as follows.

Step 1. Mark all the nodes unvisited except the beginning node s_0. Define $d(s_k,p)$ which means the tentative distance from s_0 to s_k passing p edges. Save the tentative distance $d(s_k,1)$ according to the equation $d(s_k,1)=C(s_0,s_k)$ where $C(s_0,s_k)$ means the weight of edge between s_0 and s_k. Save other tentative distances as infinite. Set s_1 as the current node.

Step 2. For the current node s_c, calculate all the tentative distances of its neighbor nodes using $d(s_k,p)= d(s_c,p-1)+ C(s_c,s_k)$. Update $d(s_k,p)$ if it is less than the previously recorded tentative distance passing p edges. Mark s_c as a

visited node and set the next node s_{c+1} as the current node.

Step 3. Repeat step 2 until all nodes are visited. The current tentative distance of destination node s_n passing p edges is saved in $d(s_n,p)$. Calculate the average cost of each path by dividing $d(s_n,p)$ by p and choose the minimum as the minimum average cost.

Fig. 5 shows a process of minimum average cost path selection. The original graph is Fig. 3(b). In Fig. 5(a) all tentative distances are initialized. In Fig. 5(b), s_2 and s_3 are visited and $d(s_2,2)$ and $d(s_3,2)$ are updated. In Fig. 5(c), s_3 is visited and $d(s_3,3)$ is updated. Notice that $d(s_3,2)$ is not updated because the calculated tentative distance is longer than the previous recorded distance. Finally, three average costs of $d(s_3,p)$ are compared and $d(s_3,2)/2$ is chosen as the minimum average cost.

The time and space complexity of minimum average cost path selection is a little bit higher than the Dijkstra algorithm, but it improves the performance in special cases when the characters are arranged intensively.

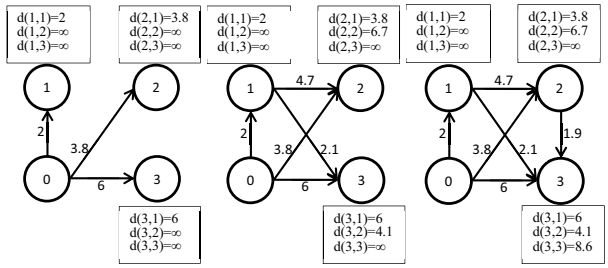


Fig. 5. Process of the minimum average cost algorithm: (a) step 1, (b) the first round of step 2 and (c) the second round of step 2.

III. EXPERIMENTAL RESULTS

A. Evaluation methods

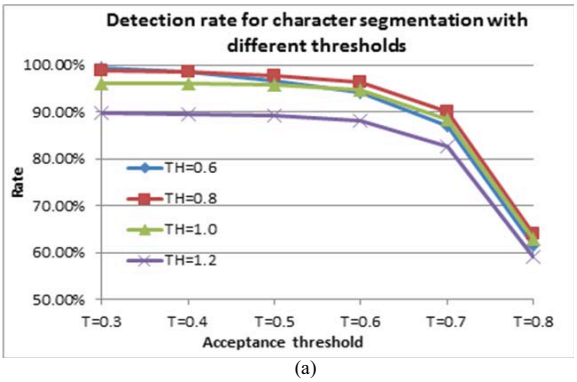
The images are binarized and segmented into text-line images by the OCR system. Afterwards the text-line images are segmented to characters by our proposed algorithm with the minimum average cost path selection. A matching score is calculated for a segmented character and a character in the ground truth.

$$Score = \frac{S(g_i \cap d_j)}{S(g_i \cup d_j)} \qquad (6)$$

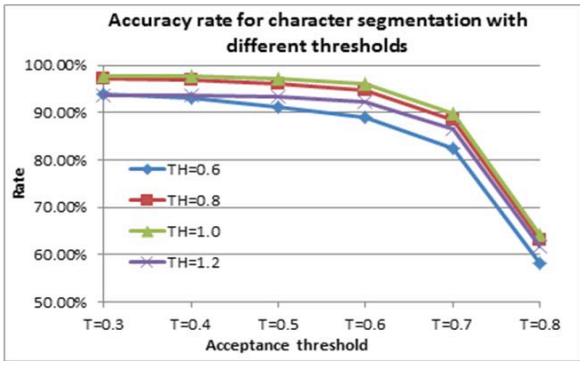
where g_i is the bounding box of a segmenting character, d_i the bounding box of a character in the ground truth and S the area of boxes. An acceptance threshold T is chosen to decide whether the two characters are matched. If Score>T, the two characters are matched, otherwise they are unmatched. The performance of segmenting result is evaluated by the detection rate and accuracy rate with different component height thresholds TH and various acceptance thresholds T.

B. Segmenting results

72 images taken from historical Dunhuang documents including 14,116 characters were tested in the experiment. The parameters α_1, α_2, β and γ in Equ. (1), (2) and (5) were set respectively according to estimation on a number of other similar historical documents. The result is shown in Fig. 6. As we can see, the detection rate reaches the maximum when component height threshold TH=0.8 and the accuracy rate reaches the maximum when TH=1. Finally the acceptance threshold is set to 0.6 due to manual evaluating experience and the height threshold TH is set to 1, where the detection rate was 94.6% and the accuracy rate was 96.1%. An example of segmenting results using our method is shown in Fig. 7. We also compared our method to a conventional projection based segmentation method, as well as our over-segmentation method without merging. Table 1 shows the result, which proves the effectiveness of our proposed method.



(a)



(b)

Fig. 6. Performance of our character segmentation algorithm with different thresholds: (a) detection rate and (b) accuracy rate.

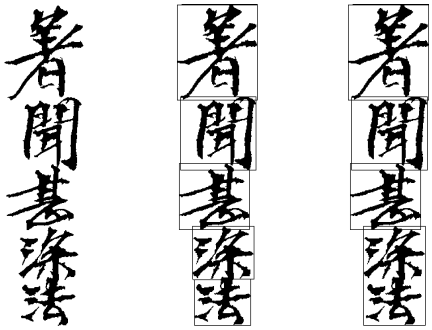
TABLE 1. COMPARISON OF OUR METHOD TO OTHERS

	Detection rate(%)	*Accuracy rate(%)*
Projection results	88.0	88.8
Over-segmentation results	90.2	71.9
Our method	94.6	96.1

Fig. 7. An example of the segmenting results of our method.

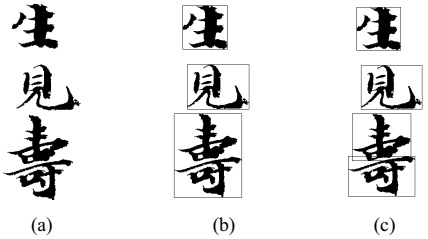
C. Error analysis

Most segmenting errors can be attributed to two reasons. The first error shown in Fig. 8 is caused by small characters connected together. The merging process merges them or the over-segmenting process overlooks them. The second error shown in Fig. 9 is caused by overlong characters. The system cuts that character to two components and decides to leave them separately. These errors are expected to be rectified by improvements of recognizer. Other errors caused by binarization or line segmentation are not presented here.



(a) (b) (c)

Fig. 8. An example of the first error: (a) a part of binary line image, (b) correct segmentation and (c) wrong segmentation.



(a) (b) (c)

Fig. 9. An example of the second error: (a) a part of binary line image, (b) correct segmentation and (c) wrong segmentation.

IV. Conclusion

In this paper, we propose a method for historical Chinese character segmentation. The proposed methodology has a remarkable performance in our samples. The experimental results prove the effectiveness of the algorithm. The method segments text-line image to characters with an over-segmenting process and a merging process. The merging process applies the minimum average cost path selection. Various geometric features and recognition feedbacks of characters are considered to calculate the costs.

Future research work will focus on two aspects, one is to develop a character recognize module with better performance on large scale historical Chinese characters. The other is improving the interacting mechanism between character segmentation and character recognition to segment touching characters.

Acknowledgment

This research is funded by National Natural Science Foundation of China (No. 61261130590, 61032008), 973 National Basic Research Program of China (No. 2014CB340500) and Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation. The authors would like to thank the National Library of China for providing historical Chinese document image samples.

References

[1] Xiaofan Lin, Xiaoqing Ding, Ming Chen, Rui Zhang, and Youshou Wu. "Adaptive confidence transform based classifier combination for Chinese character recognition." Pattern Recognition Letters19.10 (1998): 975-988.

[2] Casey, Richard G., and Eric Lecolinet. "A survey of methods and strategies in character segmentation." Pattern Analysis and Machine Intelligence, IEEE Transactions on 18.7 (1996): 690-706.

[3] Yi-Hong Tseng, and Hsi-Jian Lee. "Recognition-based handwritten Chinese character segmentation using a probabilistic Viterbi algorithm." Pattern Recognition Letters 20.8 (1999): 791-806.

[4] Min Soo Kim, Kyu Tae Cho, Hee Kue Kwag, and Jin Hyung Kim. "Segmentation of handwritten characters for digitalizing Korean historical documents." Document Analysis Systems VI. Springer Berlin Heidelberg, 2004. 114-124.

[5] Van Phan, Truyen, Bilan Zhu, and Masaki Nakagawa. "Development of nom character segmentation for collecting patterns from historical document pages." Proceedings of the 2011 Workshop on Historical Document Imaging and Processing. ACM, 2011.

[6] Nikos Nikolaou, Michael Makridis, Basilis Gatos, Nikolaos Stamatopoulos, and Nikos Papamarkos. "Segmentation of historical machine-printed documents using Adaptive Run Length Smoothing and skeleton segmentation paths." Image and Vision Computing 28.4 (2010): 590-604.

[7] Lin Yu Tseng, and Rung Ching Chen. "Segmenting handwritten Chinese characters based on heuristic merging of stroke bounding boxes and dynamic programming." Pattern Recognition Letters 19.10 (1998): 963-973.

[8] Xiaolu Sun, Liangrui Peng, and Xiaoqing Ding. "Touching character segmentation method for Chinese historical documents." IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics, 2010.

[9] Shuyan Zhao, Zheru Chi, Pengfei Shi, Qing Wang. "Handwritten Chinese character segmentation using a two-stage approach." Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on. IEEE, 2001.

23rd Internationalization and Unicode Conference (IUC23)
Unicode, Internationalization and the Web: The Global Connection
March 24-26, 2003, Prague, Czech Republic
http://www.unicode.org/iuc/iuc23/

漢字變體字: 筆謎和謎底

UniHan Variation: Issues and Solutions

Richard S. COOK
Department of Linguistics
University of California, Berkeley
rscook@socrates.berkeley.edu
http://stedt.berkeley.edu/
2003-02-11-17:30

CONTENTS

INTRODUCTION

This presentation is concerned with discussion of issues relating to solutions to the UniHan (CJKV[1] Unified Ideographs) variant problem. Some of the kinds of variation encountered in the encoded CJKV characterset are examined along with its implications for users and developers, and methods for coping with this variation in software implementations are explored. The following three aspects of the CJKV characterset are addressed:

(一) Variant Typology
(二) CJKV Description Language (CDL)
(三) Variant Selectors

Specifically, we intend to propose the use of a formal CJKV Description Language (CDL) in conjunction with Variant Selectors (VS) for resolution of CJKV variant issues. As a first step, we believe that precise descriptions using a formal CDL should be developed for all encoded CJKV entities, including unifications in which basic script elements (see the definitions below) have been deemed non-distinctive. This talk concludes with demonstration of *Wenlin 3.x* software implementing a stroke-based font for Unicode 4.0 UniHan.

[1] Chinese, Japanese, Korean, Vietnamese.

BACKGROUND

The UCS now encodes a large number of variant CJKV character forms of various types, some of which have relative mappings (i.e. traditional-to-simplified mappings), many of which do not. On the basis of IRG member submissions for CJKV Unified Ideograph Extension C1, a very large number of additional encoded variants seem imminent.

Non-systematic encoding of variant character forms presents a great danger to a character encoding, great difficulties to the maintainer and publisher of the standard, great problems for the implementer, and confusion to the end-user. Where unassimilated, non-distinctive character forms creep into the character set, all encoded variants, rare or not, appear to have an equal footing, and information access and exchange are impeded as a result.

One might wonder, "Why has it been permitted to encode character variants in the UCS in the first place, if this is such a problem?" In fact, every character has a story, and different encoded characters have different stories. Sometimes a given variant has been encoded because of legacy encoding issues (*e.g.* Compatibility characters), sometimes because of character property issues (*e.g.* Radicals), sometimes because of irregularities in the traditional organizational systems (Radical/Stroke indices), sometimes because of limitations of the rules and procedures governing the modern encoding process (*e.g.* ISO/IEC 10646 Annex S).

As the previous examples might suggest, the processes driving the encoding of variants are numerous and varied themselves. These processes point to logical classes of variants, and it is to examination of some of these that we should turn, after briefly defining "variant".

VARIANTS

Most broadly, underlying Unicode's minimalist approach to character encoding is something known as the Character-Glyph Model (CGM). In accordance with the implications of the CGM (see ISO/IEC TR 15285-1998E) one seeks to define the optimum set of script elements necessary for the encoding of a given script, based on identification of distinctive features. As historical linguistics tells us, what is distinctive in one time may not be distinctive in another: fine distinctions develop and blur and disappear over time. And so it is that as the Unicode Standard seeks to address the encoding of ancient texts, it is the very locale- and time-dependence of this key notion "distinctive feature" which presents character encoding with its greatest challenge.

What might "distinctive features" mean for Han? What exactly is a Han variant? Let's look at the second question first. Consider, for a rather vexing example, the ten forms in the following list (excerpted from line 3767 of my SWJZZ variant mapping table):

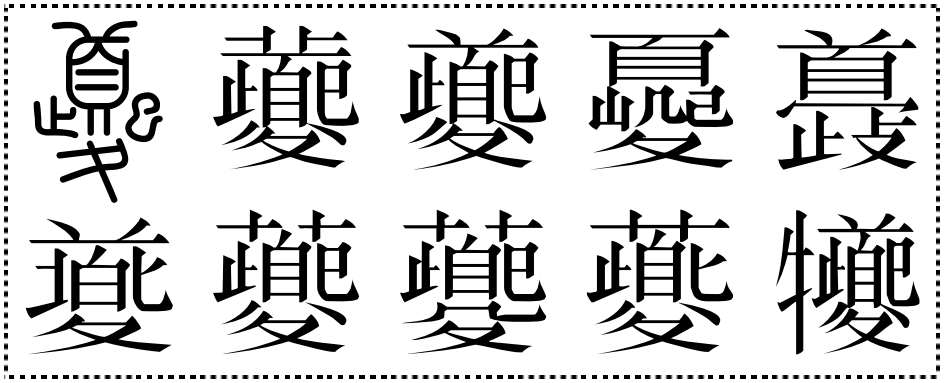


FIGURE 1: *Some Encoded Variants*

Of these, all but the first (a "Small Seal" form in Supplementary Private Use) represent actual encoded characters, as the UCS codepoints listed in FIGURE 2 illustrate:

[U+100eb6]	[U+21582]	[U+5914]	[U+2157f]	[U+2982e]
[U+21578]	[U+270f0]	[U+2f9b2]	[U+270cd]	[U+72aa]

FIGURE 2: *UCS Codepoints for FIGURE 1*

Disregarding certain attributes as non-distinctive (we will discuss what this might mean below), there is a sense in which all of the above forms represent the "same Chinese character", and it is for this reason that FIGURE 1 bears the title (really only an opinion) "Some Encoded Variants". The (121AD) lexicon *Shuowen* says that our Seal form [U+100eb6] is a writing of the name of a malevolent freakish mountain monster, 'like a dragon, but with one foot, horns, hands, and a man's face'. In early inscriptional forms the creature also had claws, a tail, and walked (hopped?) upright; not to be confused with 夔 [U+5912] (猱 [U+7331]). All of the forms in FIGURE 1 stand for (or are derived from, in the case of [U+72aa]) writings of the "mountain monster" syllable, pronounced **kuí** in modern Mandarin.

All forms in FIGURE 1 appear in *Hanyu Da Zidian* (HDZ), a primary lexical source for the IRG's Ext B encoding work, and the nine encoded forms all have (non-PUA) HDZ mappings. Of these nine, all but [U+72aa] (the last listed, with the "Ox" radical on the left) may be regarded as true "graphical variants" of one another (this term is discussed below). In a perfect world, those eight forms now enshrined separately at eight distinct codepoints (only one of which is in the BMP)

	[U+21582]	[U+5914]	[U+2157f]	[U+2982e]
[U+21578]	[U+270f0]	[U+2f9b2]	[U+270cd]	

FIGURE 3: *Eight "retroactive unifications"*

would all have been unified, in strict adherence to the CGM.

The fact that application of the (informative) unification "rules" of ISO/IEC 10646-1 Annex S did *not* result in unification of these eight forms indicates some limitations in those rules, or an over-riding "lexical source separation" rule. In fact, such an over-riding rule is implicit in the Annex S unification rules, which are based on (among other things) such traditional lexical indexing conventions as *stroke types* and *stroke counts*, to the exclusion of "meaning". If one were to insist that the above eight glyphs are all unifiable, one would have to base this opinion on the lexical definitions of each character in a particular lexical source. And different sources might disagree.

VARIANT TYPOLOGY: THE CJKV GRAPHICAL VARIANT

As exclusion of [U+72aa] from the "retroactive unification" list in FIGURE 3 illustrates, variants among the encoded and unencoded CJKV characters may be classed into several types, and not all variant types are relevant to the issues which must be resolved for UCS encoding. In fact, the only kind of variant relevant to UCS encoding is the so-called "CJKV Graphical Variant", and so we should take a moment to try to define this and a few terms relevant to the CKJVUI repertoire.

> • DEFINITION A: ***CJKV Graphical Variants*** (CGV) may here be defined as those entities which are (by consensus or significant opinion) "in some sense the same character" (non-distinctive within a particular usage context), and yet which differ from one another in terms of *basic script elements*.
>
> • DEFINITION B: ***Basic script elements*** such as *stroke type, stroke count, component structure*, or *component position* are those script elements which are usually distinctive in a specific context. In the case of CJKV Graphical Variants, this usual distinctiveness is being over-ridden, such that normally distinctive elements are viewed as non-distinctive for the purpose of building unification/variant classes.
>
> • DEFINITION C: ***Non-basic script elements***, such things as *stroke weight, skew angle, flourish, ink color, resolution*, are those elements which are *never* distinctive, and so can never be used as the basis for disunification.
>
> • DEFINITION D: ***Glyphic Variants*** are therefore distinct from Graphical Variants in that the former involve variation in non-basic script elements.
>
> • DEFINITION E: ***Script entities*** include *basic stroke types* and all complex assemblages of stroke primitives, from dependent *component* up to the level of independent CJKV *character*. [Definitions of stroke primitive, component, and character are given below.]

In terms of the CGM, it appears therefore that the CJKV Graphical Variant is a script entity falling somewhere *between* "character" and "glyph", and this suggests that a CVGM "Character-Variant-Glyph Model" is needed to address CJKV Graphical Variants.

In terms of the unification procedures described in Annex S, the members of a CGV class are sometimes identified and unified, though not always. Disunifications such as those listed in FIGURE 3 result at least in part from the lack of a mechanism for treating the graphic/glyphic difference in some meaningful and consistent way. Similarly, unifications involving glyphs with different stroke counts gloss over distinctions which may be meaningful for certain purposes (*e.g.* for producing a Radical/Stroke index in which real stroke counts for real glyphs must appear).

Within an instance of this "CJKV Graphical Variant" class, one may define a subset including all forms subject to the rules of unification: some forms in this subset may have been unified already, and some may not. One is therefore obliged to clearly state one's unification rules, and make these rules *normative*, if one is to classify forms for the purpose of unification and identifying variants.

FIGURE 4: *Members of the **áo** Graphical Varclass*

In FIGURE 4 above are listed 3 members of the **áo** Graphical Variant Class. This independent character is also a rather productive component in the script, as seen in FIGURE 5 below.

FIGURE 5: *Encoded characters with the **áo** component*

One problem is that in the forms above the solid line in FIGURE 5 the **áo** component sometimes has 10 strokes, and other times has 11 strokes, reflecting IRG unification of the forms given as [U+e109] (PUA) and [U+6556] (BMP) in FIGURE 4 above. That is, variant forms with either 10 or 11 strokes have been unified. To find one of these characters in a Radical/Stroke index, one would not know whether to look under 10 or 11 residual strokes, and the form found in a Unicode R/S index under either of these stroke counts might have one or the other actual stroke counts.

It has sometimes been suggested that one might like to unify, for example, all regular traditional/simplified pairs (that is, cases where "simplified" forms are not "traditional"). The rules for determining regular relations and also for treating exceptions among the distinctive elements of these pairs would need to be formulated. A great deal of consensus on such matters has already been achieved in PRC work. FIGURE 6 below lists some standard PRC Traditional/Simplified pairings. [Roughly, the characters in ROW B are simplified (to A) only as independent characters, never as components. The characters in ROW D are the opposite, simplified (to C) only as components, but never as independent characters, except for the last.]

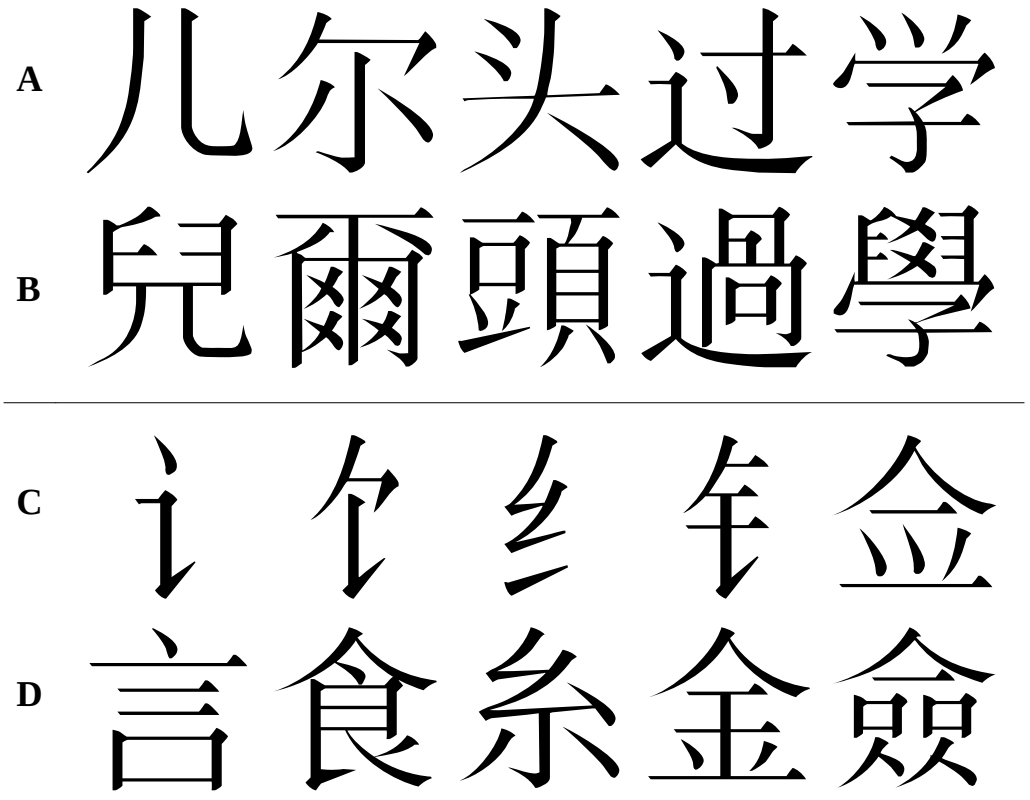
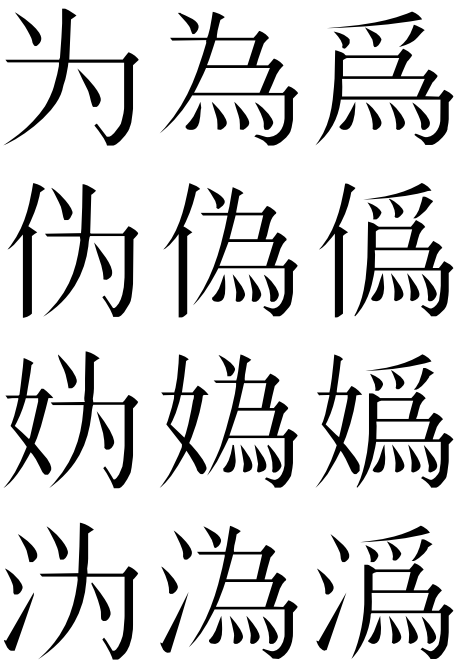


FIGURE 6: *Some Traditional/Simplified pairs*

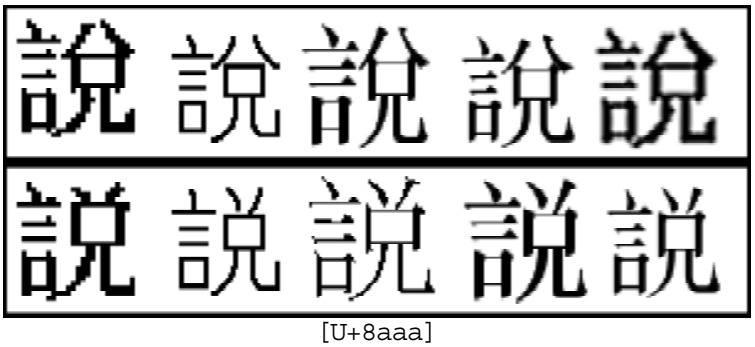
It is arguable, however, that the bottom S/T pairs in FIGURE 6 are suitable for treatment with Variant Selectors (as dependent Graphical Variants), while most upper pairs are not. But since all characters in FIGURE 6 are encoded, including the Graphical Variants (some in the "Radical" block), any scheme for treating variation in these pairs must be handled with mapping tables (see below).



[U+4e3a] [U+70ba] [U+7232]
[U+4f2a] [U+507d] [U+50de]
[U+59ab] [U+5aaf] [U+5b00]
[U+6ca9] [U+6e88] [U+6f59]

FIGURE 7: *Encoded "Triplets"*

FIGURE 7 above lists 4 sets of encoded "Triplets". The first column from the left is the "simplified" form, the second form is the "full" Big-5 form, the third is the "old" a.k.a. "ultra-fanti" form.



[U+8aaa]
[U+8aac]

FIGURE 8: *Glyphic vs. Graphical Variation*

FIGURE 8 above is intended to illustrate the difference between *Glyphic Variants* and *Graphical Variants*. The top row lists *glyphic variants* of [U+8aaa], while the bottom row lists *glyphic variants* of [U+8aac]. The characters [U+8aaa] and [U+8aac] are however *Graphical Variants*, in that they differ from one another in *basic script elements* (*i.e.*, basic stroke types).

WHY A CJKV DESCRIPTION LANGUAGE (CDL)?

Why do we need to define a formal CJKV Description Language (CDL) and why apply it to the entire CJKV characterset? What would an adequate CDL do for us? What about other systems such as Unicode's IDC/IDS? Isn't there some existing system that can be applied to the problem? Do we need to invent a new system?

Let's look at these questions here, one at a time.

• First, the reasons we need to define a formal CJKV Description Language (CDL) and apply it to the entire CJKV characterset relate to resolution of the Han Variant Problem. In the CGM, one is content to leave the exact form of the character underspecified. In a "CVGM", one must exactly specify the shapes of all encoded entities, since CJKV Graphical Variants are now to be treated as encoded entities using variant selectors.

• An adequate CDL would provide us with reference glyphs for all encoded entities, storing a great deal of important information in the reference glyphs themselves. Things like "lexical radical", "component structure", "stroke count" and "stroke order" are all specified by the reference glyph.

• Incidentally, though quite significantly, an adequate CDL would provide us with the necessary information to automatically generate computer fonts for production and promulgation of the standard.

• Unicode's IDC/IDS (Ideographic Descriptive Characters and Sequences) are acknowledged as only a glance in the direction of a complete CDL. The IDC/IDS were never intended to serve the purpose here envisaged for a complete CDL. Other such systems, for example, the Big5-based system developed in the InfoTech Lab at Academia Sinica (see my encoding proposal "02221-n2480.pdf" which lays out the system of "operators" used in that system), are rather similar, in that they do not define the necessary basic script elements for a complete description of all Han characters.

• Wenlin Software's highly refined C implementation of its "Stroking Instruction Language" is to our knowledge the only existing, fully implemented, completely adequate CJKV Description Language. Using this language, "Stroking Instructions" have been created for some 30,000 CJKV entities (at last count), both in the BMP and in the SIP, both encoded and unencoded (PUA). It is planned at present to elaborate elements of this system such as may prove relevant to the standardization in future encoding proposals. Without going into the exact details of the specific implementation, we may however sketch the basic outlines of the elements of such a system here.

ELEMENTS OF THE CJKV DESCRIPTION LANGUAGE (CDL)

An adequate CJKV Description Language must be able to describe all uniquely CJKV script entities. The elements of an adequate CJKV Description Language are to be specified in terms of a grid of specific size, and in terms of the entities which may populate that grid. These entities, here termed "Basic Stroke Types", are defined as follows.

> • DEFINITION F: ***Basic Stroke Types***, also called *stroke primitives*, are the finite though extensible set of fundamental script entities, each composed of precisely one "stroke" (uninterrupted movement of the brush or pen), as "stroke" is defined in the traditional Song writing style employed in primary modern sources of lexical stroke-based indices. Each Basic Stroke Type is defined in the CDL in terms of a set of *Segment Types*. Each Segment Type is defined in terms of a set of *Points*, and each Point is defined in terms of a set of Cartesian *Coordinates*. Cf. FIGURE 9.

FIGURE 9: *Selected Basic Stroke Types (arranged according to shape)*

FIGURE 9 lists some representative Basic Stroke Types selected from the currently defined set of approximately 50.

Using a set of Basic Stroke Types, complete descriptions of any CJKV script entity may be built. It is anticipated that a complete inventory of basic stroke types for the CJKV elements of the Unicode 4.0 characterset might require a block of about 64 codepoints. Several rare stroke types are known to us at present, though as yet unsupported.

Certain other classes of CJKV entity require that transformations, rather than additional stroke types, be defined in the CDL. The top row of FIGURE 10 below lists some of these; the bottom row gives forms with usual stroke types:

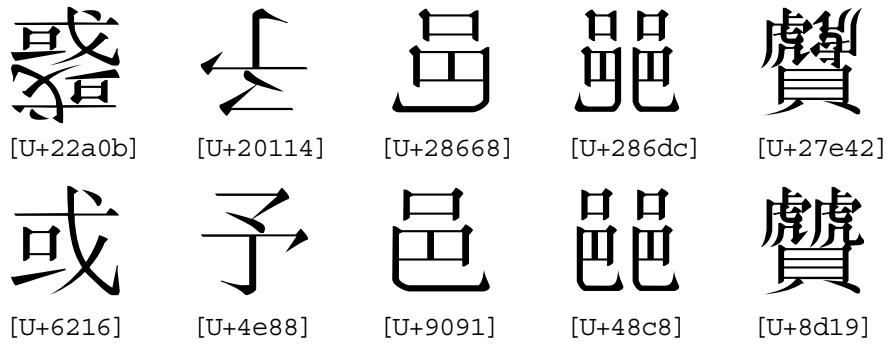


FIGURE 10: *Unusual stroke types in "flipped" components*

The forms given in FIGURE 10 all involve components in which a Basic Stroke Type has been written in some non-standard way, which is to say, the stroke is the inversion or mirror image of a basic stroke type. Other rare stroke types are seen in encoded "cursive" forms: the correct way to treat non-Kaishu forms must be determined. It seems that the best solution involves "kaishu-ification" of such cursive forms. The current reference glyph (on the left in FIGURE 11 below) for [U+201ad] involves elements of this type (kaishu-ified using basic stroke types, on the right).

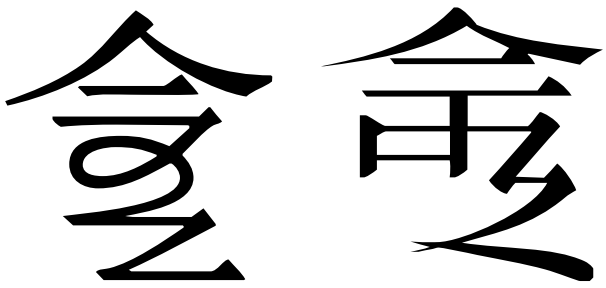


FIGURE 11: *[U+201ad]: Unicode 3.2 reference glyph, and its "kaishu-ification"*

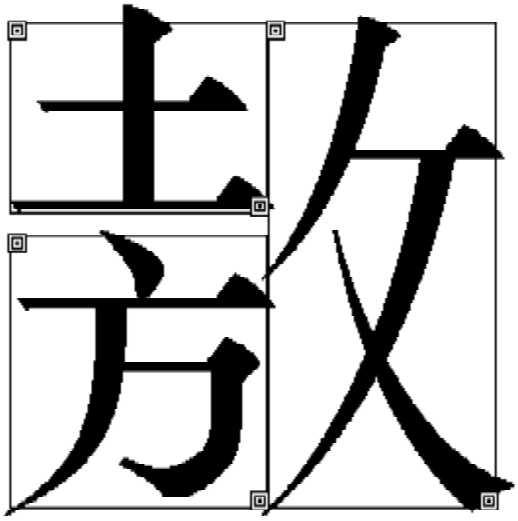


FIGURE 12: *CDL for PUA graph [U+e109]*

The CDL description of `[U+e109]` (PUA graphical variant of `[U+6556]`; cf. FIGURE 4 above) specifies the components `[U+571f]`, `[U+65b9]`, and `[U+6535]`, each with the (x, y) coordinates of its bounding rectangle within the bounding square of the composite character. This description is not self-contained: in order to display the composite character, the language interpreter uses the CDL descriptions of each of the three components. In general, components can be any characters that are themselves defined as sequences of basic strokes and/or simpler components.

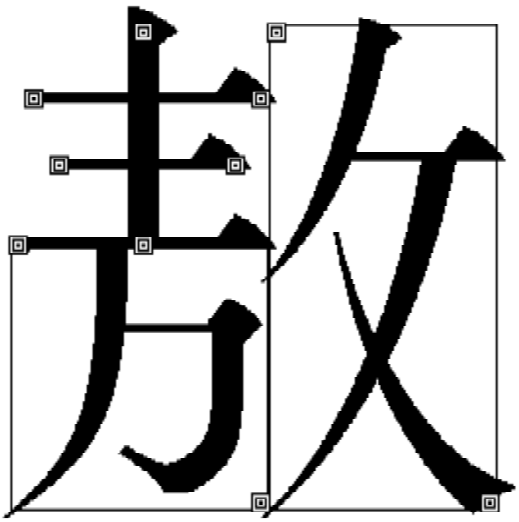


FIGURE 13: *CDL for BMP graph [U+6556]*

The CDL description of `[U+6556]` specifies the first three strokes as basic stroke types, each with the coordinates of its starting and ending points (and possibly stroke modifiers), and then specifies the two components `[U+4e07]` and `[U+6535]` with their bounding rectangles.

VARIANT SELECTORS (VS)

Unification classes (as these have been set up for the CJKVUI so far) are simply special cases of *variant classes* (which remain to be established). With unification classes, the variation among the class members has in some regard been deemed irrelevant to encoding, and in effect discarded from standardization (as abandonment of the multi-columnar format for the printing of Ext. B illustrates). As discussed above, there are indeed some reasons to reconsider the ultimate utility of a unification model which does not distinguish basic and non-basic script elements, and which does not standardize (preserve) such information.

In the model which we would put forth for application of variant selectors to CJKV encoding, variant classes would be comprised of all unified (or unifiable) forms involving basic script elements, both encoded and unencoded. That is, unified entities distinguishable in terms of basic script elements (e.g. stroke count) would in effect be disunified employing the variant selector scheme. Script entities suitable for encoding in the VS scheme would be all basic script entities (as these were defined above), both independent and dependent, including contextually dependent (that is, compositionally deformed) forms. It is argued that the distinction between independent and dependent basic script elements is not relevant to encoding. Rather, "graphical component" is defined as follows:

> • DEFINITION G: ***Graphical Components*** are any productive assemblage of basic stroke types, where productivity is gauged in terms of frequency of occurrence. That is, a component is identified in terms of its appearance (as a recurrent pattern of basic stroke types) in two or more non-unified glyphs. A *Graphical Component* may or may not be encoded, depending upon its degree of productivity (or importance, according to some authoritative source). Highly productive or highly salient components should most definitely be encoded immediately. Less productive components may be encoded on an as-needed basis.

In application of this Variant Selector scheme to CJKV script entities, use of VS is completely optional, and so we foresee two kinds of text: text *not* employing variant selectors ("text"); text employing variant selectors ("VS text"). In VS text every basic script entity would be identified in the form

`[U+X]([U+V])? => <CDL>`

FIGURE 14: *Variant Selector scheme for CJKV script entities*

where "[U+X]" represents a CJKV character codepoint, ([U+V])? indicates the optional Variant Selector, <CDL> indicates that there is a precise description (stroking instructions) of the entity using the CDL, and "=>" indicates the mapping relation between the variant selector/codepoint pair and the <CDL>. Though the VS is optional, any block of VS text will necessarily include at least one VS. In this scheme, all script entities in VS text are treated as variants. In cases of VS text in which the VS is not used, that is, in cases in which a variant form has not been explicitly chosen, the system would be expected to fall back to some locale-specific default among the variants associated with a particular Unicode Scalar Value.

Why use Variant Selectors at all, rather than simply continue encoding variants? Why not simply develop and maintain variant mappings as needed in Unihan.txt? There are several reasons why VS might be viewed as preferable to simple mapping tables, though we do not believe that VS and

mapping tables are mutually exclusive. On the contrary, VS *and* variant mappings will both be necessary to handle encoded and unencoded variants. A VS scheme simplifies searching, since search algorithms do not fail in a VS scheme with addition of variants: the VS provides an extensible means of dealing with variants, whereas the non-VS mapping table approach requires mapping table maintenance without an adequate organizational mechanism. Most importantly, it is the tying together of VS with CDL which provides the most important long-term benefits. Variant tables would also use the VS scheme:

```
([U+X]([U+V]) => <CDL>) => ([U+Y]([U+V]) => <CDL>)
```

FIGURE 15: *Variant Selectors in Variant Mapping*

SUMMARY

In summary, it may be said that we intend to propose the use of a formal CJKV Description Language (CDL) in conjunction with Variant Selectors (VS) for resolution of CJKV variant issues. Ultimate resolution of variant issues will require that the unification procedures outlined in Annex S be refined and made normative. As a first step, we believe that precise descriptions using a formal CDL should be developed for all encoded CJKV entities, including unifications in which basic script elements (see the definitions above) have been deemed non-distinctive. Such unifications would then be the initial test cases for application of VS to CJKV.

CONCLUSION

As might be suspected from this brief discussion, the "Han variant problem" is deep and difficult, and will not be resolved quickly. On the contrary, seeking to solve the problem without first working out a careful methodology built on well-defined principles must only compound future difficulties. Since the characters derive from multiple lexical sources, and since these sources are not always in agreement, the UCS is now a new super-lexical source. One may view the encoded characterset in its present state as the first adequate statement of the problem. One is obligated now to take this mass of partially assimilated or completely unassimilated data and evaluate it scientifically, for the purpose of refining the method to be used in future encoding work.

ACKNOWLEDGMENTS

This paper paper was composed in association with Thomas BISHOP <wenlin@wenlin.com>, inventor of the CDL discussed here. Thanks to Tom for prescient thinking, indefatigable programming, numerous suggestions, and for providing text for the discussion of FIGURES 12 and 13. All figures containing CJKV strokes and graphs were produced using Wenlin software; descriptions of Wenlin's CDL are used here by permission. Detailed examples of Wenlin's CDL notation are excluded from this paper, pending publication. For more information on Wenlin software, please visit <http://www.wenlin.com/>.

The writing of this paper was supported in part by STEDT grants from:

- The National Science Foundation (NSF), Division of Behavioral & Cognitive Sciences, Linguistics, Grant No. BCS-9904950;
- The National Endowment for the Humanities (NEH), Preservation and Access, Grant No. PA-23353-99.

Rank-frequency relation for Chinese characters

W.B. Deng,[1,2,3] A.E. Allahverdyan,[1,4,*] B. Li,[5] and Q. A. Wang[1,3]

[1] *Laboratoire de Physique Statistique et Systèmes Complexes, ISMANS, 44 ave. Bartholdi, Le Mans 72000, France*
[2] *Complexity Science Center and Institute of Particle Physics, Hua-Zhong Normal University, Wuhan 430079, China*
[3] *IMMM, UMR CNRS 6283, Université du Maine, 72085 Le Mans, France*
[4] *Yerevan Physics Institute, Alikhanian Brothers Street 2, Yerevan 375036, Armenia*
[5] *Department of Chinese Literature, University of Heilongjiang, Harbin 150080, China*

We show that the Zipf's law for Chinese characters perfectly holds for sufficiently short texts (few thousand different characters). The scenario of its validity is similar to the Zipf's law for words in short English texts. For long Chinese texts (or for mixtures of short Chinese texts), rank-frequency relations for Chinese characters display a two-layer, hierarchic structure that combines a Zipfian power-law regime for frequent characters (first layer) with an exponential-like regime for less frequent characters (second layer). For these two layers we provide different (though related) theoretical descriptions that include the range of low-frequency characters (hapax legomena). The comparative analysis of rank-frequency relations for Chinese characters versus English words illustrates the extent to which the characters play for Chinese writers the same role as the words for those writing within alphabetical systems.

PACS numbers: 89.75.Fb, 89.75.Da, 05.65.+b

I. INTRODUCTION

Rank-frequency relations provide a coarse-grained view on the structure of a text: one extracts the normalized frequencies of different words $f_1 > f_2 > ...$, orders them in a non-increasing way and studies the frequency f_r as a function of its rank r. One widely known aspect of this rank-frequency relation that holds for texts written in many alphabetical languages is the Zipf's law; see [1–4] for reviews, [5–8] for modern instances of the law, and [9] for extensive lists of references on the subject. This regularity was first discovered by Estoup [10]:

$$f_r \propto r^{-\gamma} \quad \text{with} \quad \gamma \approx 1. \tag{1}$$

The message of a power-law rank-frequency relation is that there is no a single group of dominating words in a text, they rather hold some type of hierarchic, scale-invariant organization. This contrasts to the exponential-like form of the rank-frequency relation that would display a dominant group of words that is representative for the text.

The simple form of the Zipf's law hides the mechanism behind it. Hence there is no consensus on the origin of the law, as witnessed by different theories proposed to explain it [11–17]. An influential group of theories explain the law from certain general premises of the language [11–14], e.g. that the language trades-off between maximizing the information transfer and minimizing the speaking-hearing effort [11], or that the language employs its words via the optimal setting of information theory [12]. The general problem of derivations from this group is that explaining the Zipf's law for the language (and verifying it for a frequency dictionary) does not yet mean to explain the law for a concrete text, where the frequency of the same word varies widely from one text to another and is far from its value in a frequency dictionary.

It was held once that the Zipf's law is not especially informative, since it is recovered by very simple stochastic models, where words are generated through random combinations of letters and space symbol seemingly reproducing the $f_r \propto r^{-1}$ shape of the law [15]. But the reproduction is elusive, since the model is based on features that are certainly unrealistic for natural languages, e.g. it predicts a huge redundancy (many words have the same frequency and length) [18]. More recent opinions reviewed in [19] indicate that the Zipf's law *is* informative and *not* reducible to any trivial statistical regularity. These opinions are confirmed by a recent derivation of the Zipf's law from the ideas of latent semantic analysis [17]. The derivation accounts for generalizations of the Zipf's law for high and low frequencies, and also describes (simultaneously with the Zipf's law) the hapax legomena effect [1]; see Appendix A for the glossary of the used linguistic terms.

However, the Zipf's law was so far found to be absent for the rank-frequency relation of Chinese characters [20–25], which play—sociologically, psychologically and (to some extent) linguistically—the same role for Chinese

*Email: armen.allahverdyan@gmail.com

[1] Hapax legomena means literally the set of words that appear in the text only once. We shall employ this term in a broader sense as the set of words that appear few times, so that sufficiently many words have the same frequency. The description of this set is sometimes referred to as the frequency spectrum.

readers and writers as the words do in Indo-European languages [26–28].

Rank-frequency relations for Chinese characters were first studied by Zipf and coauthors who did not find the Zipf's law [29]. They claimed to find another power law with exponent $\gamma = 2$ [29], but this result was later on shown to be incorrect [21], since it was not based on any goodness of fit measure. It was also proposed that the data obtained by Zipf are reasonably fit with a logarithmic function $f_r = a + b\ln(c + r)$ with constant a, b and c [21]. The result on the absence of the Zipf's law was then confirmed by other studies [22–25, 30]. All these authors agree that the proper Zipf's law is absent (more generally a power law is absent), but have different opinions on the (non-power-law) form of the rank-frequency relation for Chinese characters: logarithmic [21], exponential $f_r \propto e^{-dr}$ (where $d > 0$ is a constant) [22–24, 30] or a power-law with exponential cutoff [20, 25]. In [31], the authors describe two different classes of rank-frequency relations for English and Chinese literacy works, they also proposed a model to generate such different situations.

The Zipf's law is regarded as a universal feature of human languages on the level of words [32] [2]. Hence the invalidity of the Zipf's law for Chinese characters has contributed to the ongoing debate on controversies (coming from linguistics and experimental psychology) on whether and to which extent the Chinese writing system is similar to phonological writing systems [36–38]; in particular, to which extent it is based on characters in contrast to words [3].

Results reported in this work amount to the following:

– The Zipf's law holds for sufficiently short (few thousand different characters) Chinese texts written in Classic or Modern Chinese [4]. Short texts are important, because they are building blocks for understanding long texts. For the sake of rank-frequency relations, but also more generally, one can argue that long texts are just mixtures (joining) of smaller, thematically homogeneous pieces. This premise of our approach is fully confirmed by our results.

– The validity scenario of the Zipf's law for short Chinese texts is basically the same as for short English texts [5]: the rank-frequency relation separates into three ranges. *(1)* The range of small ranks (more frequent characters) that contains mostly function characters; we call it the pre-Zipfian range. *(2)* The (Zipfian) range of middle ranks (more probable words) that contains mostly content characters. *(3)* The range of rare characters, where many characters have the same small frequency (hapax legomena).

– The essential difference between Chinese characters and English words comes in for long texts, or upon mixing (joining) different short texts. When mixing different English texts, the range of ranks where the Zipf's law is valid quickly increases, roughly combining the validity ranges of separate texts. Hence for a long text the major part of the overall frequency is carried out by the Zipfian range. When mixing different Chinese texts, the validity range of the Zipf's law increases very slowly. Instead there emerges another, exponential-like regime in the rank-frequency relation that involves a much larger range of ranks. However, the Zipfian range of ranks is still (more) important, since it carries out some 40% of the overall frequency. This overall frequency of the Zipfian range is approximately constant for all (numerous and semantically very different) Chinese texts we studied.

– We describe these two regimes via different (though closely related) theories that are based on the recent approach to rank-frequency relations [17]. This description includes a rather precise theories for rare characters (hapax legomena range) both for long and short Chinese texts.

This work is organized as follows. The next section gives a short introduction to Chinese characters and their differences and similarities with English words. Section III uncovers the Zipf's law for short Chinese texts and compares it with the English situation. Section IV studies the fate of the Zipf's law for long Chinese texts. We summarize in the last section. Appendix A contains the glossary of the used linguistic terms. Appendix B refers to the interference experiments distinguishing between Chinese characters and English words. Appendix C recollects information on the studied Chinese texts. Appendix D lists the key-characters of one studied modern Chinese text. Appendix E reminds the Kolmogorov-Smirnov test that is employed for checking the quality of our numerical fitting.

[2] Applications of the Zipf's law to automatic keyword recognition are based on this fact [33], because keywords are located mostly in the validity range of the Zipf's law. A related set of applications of this law refers to distinguishing between artificial and natural texts, fraud detection [34] *etc*; see [35] for a survey of applications in natural language processing.

[3] We stress already here that the Zipf's law holds for Chinese [25] and Japanese words [39]. This is expected and intuitively follows from the possibility of literal translation from Chinese to English, where (almost) each Chinese word is mapped to an English one (see our glossary at Appendix A for definition of various special terms). In this sense, the validity of the Zipf's law for Chinese words is consistent with the validity of this law for English texts.

[4] The Modern Chinese texts we studied are written with simplified characters, while our Classic Chinese texts are written with traditional characters. Reforms started in the mainland China since late 1940's simplified about 2235 characters. Traditional characters are still used officially in Hong-Kong and Taiwan.

[5] Here and below we refer to a typical Indo-European alphabetical based language as English, meaning that for the sake of the present discussion differences between various Indo-European and/or Uralic languages are not essential. Likewise, we expect that the basic features of the rank-frequency analysis of Chinese characters will apply for those languages (e.g. Japanese), where the Chinese characters are used.

II. CHINESE CHARACTERS VERSUS ENGLISH WORDS

Here we shortly remind the main differences and similarities between Chinese characters and English words; see Footnote 5 in this context. This subject generated several controversies (myths as it was put in [37]), even among expert sinologists [27, 28, 36–38, 40].

This section is not needed for presenting our results (hence it can be skipped upon first reading), but is necessary for a deeper understanding of our results and motivations.

The main qualitative conclusion of this section is that in contrast to English words, Chinese characters have generally more different meanings, they are more flexible, they could combine with other characters to convey different specific meanings. So there are characters, which appear many times in the text, but their concrete meanings are different in different places.

1. The unit of Chinese writing system is the character: a spatially marked pattern of strokes phonologically realized as a single syllable (please consult Appendix A for a glossary of various linguistic terms used in the paper). Generally, each character denotes a morpheme or several different morphemes.

2. The Chinese writing system evolved by emphasizing the concept of the character-morpheme, to some extent blurring the concept of the multi-syllable word. In particular, spaces in the Chinese writing system are put in between of characters and not in between of multi-syllable words [6]. Thus a given sentence can have different meanings when being separated into different sequences of words [40], and parsing a string of Chinese characters into words became a non-trivial computational problem; see [43] for a recent review.

3. Psycholinguistic research shows that the characters are important cognitive and perceptual units for Chinese writers and readers [26–28], e.g. Chinese characters are more directly related to their meanings than English words to their meanings [28] [7]; see Appendix B for additional details. The explanation of this effect would be that characters (compared to English words) are perceived holistically as a meaning-carrying objects, while English words are yet to be reconstructed from a sequence of their constituents (phonemes and syllables) [8].

4. One-character words dominate in the following specific sense. Some 54% of modern Chinese word *tokens* are single-character, two-character word tokens amount to 42%; the remaining words have three or more characters [45]. For modern Chinese word *types* the situation is different: single character words amount to some 10% against 66% of two-character words [45]. Classic Chinese texts have more single-character words (tokens), the percentage varies between some 60% and 80% for texts written in different periods.

The modern Chinese has $\approx$ 10440 basic (root) morphemes. 93 % of them are represented by single characters. The overall number of Chinese characters is $\approx$ 18000.

5. A minor part of multi-character words are multi-character morphemes, i.e. their separate characters do not normally appear alone (they are fully bound). Examples of this are the two-character Chinese words for *grape* "葡萄" *(pú táo)*, *dragonfly* "蜻蜓" *(qīng tíng)*, *olive* "橄榄" *(gǎn lǎn)*. Estimates show that some 10% of all characters are fully bound [37].

A related set of examples is provided by two-character words, where the separate characters do have an independent meaning, but this meaning is not directly related to the meaning of the word, e.g. "东西" *(dōng xī)* means *thing*, but literally it amounts to *east-west*, or "手足" *(shǒu zú)* means *close partnership*, but literally *hand-foot*.)

6. The majority of the multi-character words are semantic compounds: their separate characters can stand alone and are related to the overall meaning of the word. Importantly, in most cases, the separate meanings of the component characters are wider than the (relatively unique) meaning of the compound two-character word. An example of this situation is the two-character Chinese word for *train* "火车" *(huǒ chē)*: its first character "火" *(huǒ)* has the meaning of *fire, heat, popular, anger, etc*, while the second character "车" *(chē)* has the meaning of *vehicle, machine, wheeled, lathe, castle, etc.*

Note that in Chinese there is a certain freedom in grouping morpheme into different combinations. Hence it is not easy to distinguish the semantic compounds from lexical phrases.

7. At this point we shall argue that in general Chinese characters have a larger number of different meanings than English words. This statement will certainly appear controversial, if it is taken without proper caution, and is explained without proper usage of linguistic terms (see our glossary at Appendix A); consult Footnote 12 in this context.

---

[6] An immediate question is whether Chinese readers will benefit from reading a character-written text, where the words boundaries are indicated explicitly. For normal sentences the readers will not benefit, i.e. it does not matter whether the word boundaries are indicated explicitly or not [41]. But for difficult sentences the benefit is there [42].

[7] To get a fuller picture of this effect let us denote $\tau_f(E)$ and $\tau_f(C)$ for English and Chinese phonology activation times, respectively, while $\tau_m(E)$ and $\tau_m(C)$ stand for respective meaning activation times. The phonology activation time is the time passed between seeing a word in English (or character in Chinese) and pronouncing it; likewise, for the meaning activation time. Now these quantities hold [28]: $\tau_f(E) < \tau_m(C) \simeq \tau_f(C) < \tau_m(E)$.

[8] A simpler explanation would be that the characters are perceived as pictograms directly pointing to their meaning. In its literal form this explanation is not correct, since characters-pictograms are not frequent in Chinese [37, 44].

First of all note the difference between polysemes and homographs: polysemes are two related meanings of the same character (word), homographs are two characters (words) that are written in the same way, but their meanings are far from each other [9]. Now many characters are simultaneously homographs and polysemes, e.g. character "明" *(míng)* means *brilliant, light, clear, next, etc.* Here the first three meanings are related and can be viewed as polysemes. The fourth meaning *next* is clearly different from the previous three. Hence this is a homograph. Another example is the character "发" *(fā or fà)* that can mean *hair, send out, fermentation, etc.* All these three meanings are clearly different; hence we have homographs. Note the following peculiarity of the above two examples: the first example is a non-heteronym (homophonic) character, i.e. it is read in the same way irrespectively whether it means *light* or *next*. The second example is a heteronym character: it written in the same way, but is read differently depending on its meaning.

In most cases, heteronym characters—those which are written in the same way, but have different pronunciations—have at least two sufficiently different meanings. The disambiguation of their meaning is to be provided by the context of the sentence and/or the shared experience of the writer and reader [10].

Surely, also English words can be ambiguous in meaning (e.g. *get* means *obtain*, but also *understand = have knowledge*), but there is an essential difference. The major contribution of the meaning ambiguity in English is the polysemy: one word has somewhat different, but also closely related meanings. In contrast, many Chinese characters have widely different meanings, i.e. they are homographs rather than polysemes.

However, we are not aware of any quantitative comparison between homography of Chinese versus English. This may be related to the fact that it is sometimes not easy to distinguish between polysemy and homophony (see the glossary in Appendix A). Still the above statement on Chinese characters having a larger number of different meanings can be quantitatively illustrated via the relative prevalence of heteronyms in Chinese. The amount of heteronyms in English is negligible, e.g. in rather complete list of heteronyms presented in [46], we noted only 74 heteronyms [11], and only three of them had more than 2 meanings. This is a tiny amount of the overall number of English words ($> 5 \times 10^5$). To compare this with the Chinese situation, we note that at least some 14% of modern Chinese and 25% of traditional characters are heteronyms, which normally have at least two widely different meanings. Within the most frequent 5700 modern characters the number of heteronyms is even larger and amounts to 22 % [45] [12].

8. Chinese nouns are generally less abstract: whenever English creates a new word via conceptualizing the existing one, Chinese tends to explain the meaning via using certain basic characters (morphemes). Several basic examples of this scenario include: length=long+short "长 短" *(cháng duǎn)*, landscape=mountains+water "山 水" *(shān shuǐ)*, adult=big+person "大 人" *(dà rén)*, population=person+mouth "人 口" *(rén kǒu)*, astronomy=heaven+script "天 文" *(tiān wén)*, universe=great+emptiness "太空" *(tài kōng)*. English tools for making abstract words include prefixes, *poly-*, *super-*, *pro-*, *etc* and suffixes, *-tion*, *-ment*. These tools either do not have Chinese analogs, or their usage can generally be suppressed.

English words have inflections to indicate the tense of verbs, the number for nouns or the degree for adjectives. Chinese characters generally do not have such linguistic attributes [13], their role is carried out by the context of the sentence(s) [14].

To summarize this section, the differences between Chinese and English writing systems can be viewed in the context of the two features: emphasizing the role of base (root) morphemes and delegating the meaning to the context of the sentence whenever this is possible [26].

The quantitative conclusion to be drawn from the

[9] Note that polysemes are defined to be related meanings of the same word, while homographs are defined to be different words. This is natural, but also to some extent conventional, e.g. one can still define homographs as far away meanings of the same word.

[10] Note that homophony in Chinese is much larger than homography: in average a syllable has around 12–13 meanings [26]. Hence, in a sense, characters help to resolve the homophony of Chinese speech. This argument is frequently presented as an advantage of the character-based writing system, though it is not clear whether this system is here not solving the problem that was invited by its usage [44].

[11] Not counting those heteronyms that arise because an English word happens to coincide with a foreign special name, e.g. *Nancy* [English name] and *Nancy* city in France.

[12] One should not conclude that in average the Chinese character has more meanings than the English word, because there is a large number of characters—between 10 and 14 % depending on the type of the dictionary employed [47]—that do not have lexical meaning, i.e. they are either function words (grammatical meaning mainly) or characters that cannot appear alone (bound characters). If now the number of meanings for each character is estimated via the number of entries in the explanatory dictionary—which is more or less traditional way of searching for the number of meanings, though it mixes up homography and polysemy—the average number of meanings per a Chinese character appears to be around 1.8–2 [47]. This is smaller than the average number of (necessarily polysemic) meanings for an English word that amounts to 2.3.

[13] Chinese expresses temporal ordering via context, e.g. adding words *tomorrow* or *yesterday*, or by aspects. The difference between tense and aspect is that the former implicitly assumes an external observer, whose reference time is compared with the time of the event described by the sentence. Aspects order events according to whether they are completed, or to which extent they are habitual. Indo-European languages tie up tense and aspect. The tie is weaker for Slavic Indo-European languages. Chinese has several tenses including perfective, imperfective and neutral.

[14] Chinese has certain affixes, but they can be and are suppressed whenever the issue is clear from the context.

above discussion is that Chinese characters have more different meanings, they are flexible, they could combine with other characters to convey different specific meanings. Anticipating our results in the sequel, we expect to see a group of characters, which appear many times in the text, but their concrete meanings are different in different places of the text.

III. THE ZIPF'S LAW FOR SHORT TEXTS

We studied several Chinese and English texts of different lengths and genres written in different epochs; see Tables I, II and III. Some Chinese texts were written using modern characters, others employ traditional Chinese characters; see Tables I and II. Chinese texts are described in Appendix C. English texts are described in Table III. The texts can be classified as short (total number of characters or words is $N = 1 - 3 \times 10^4$) and long ($N > 10^5$). They generally have different rank-frequency characteristics, so discuss them separately.

For fitting empiric results we employed the linear least-square method (linear fitting), but the also checked its results with other methods (KS test, non-linear fitting and the maximum likelihood method). We start with a brief remainder of the linear fitting method.

A. Linear fitting

For each Chinese text we extract the ordered frequencies of different characters [the number of different characters is n; the overall number of characters in a text is N]:

$$\{f_r\}_{r=1}^n, \quad f_1 \geq ... \geq f_n, \quad \sum\nolimits_{r=1}^n f_r = 1. \tag{2}$$

Exactly the same method is applied to English texts for studying the rank-frequency relation of words.

We fit the data $\{f_r\}_{r=1}^n$ with a power law: $\hat{f}_r = cr^{-\gamma}$. Hence we represent the data as

$$\{y_r(x_r)\}_{r=1}^n, \quad y_r = \ln f_r, \quad x_r = \ln r, \tag{3}$$

and fit it to the linear form $\{\hat{y}_r = \ln c - \gamma x_r\}_{r=1}^n$. Two unknowns $\ln c$ and γ are obtained from minimizing the sum of squared errors [linear fitting]

$$SS_{\rm err} = \sum\nolimits_{r=1}^n (y_r - \hat{y}_r)^2. \tag{4}$$

It is known since Gauss that this minimization produces

$$-\gamma^* = \frac{\sum_{k=1}^n (x_k - \overline{x})(y_k - \overline{y})}{\sum_{k=1}^n (x_k - \overline{x})^2}, \quad \ln c^* = \overline{y} + \gamma^* \overline{x}, \tag{5}$$

where we defined

$$\overline{y} \equiv \frac{1}{n}\sum\nolimits_{k=1}^n y_k, \quad \overline{x} \equiv \frac{1}{n}\sum\nolimits_{k=1}^n x_k. \tag{6}$$

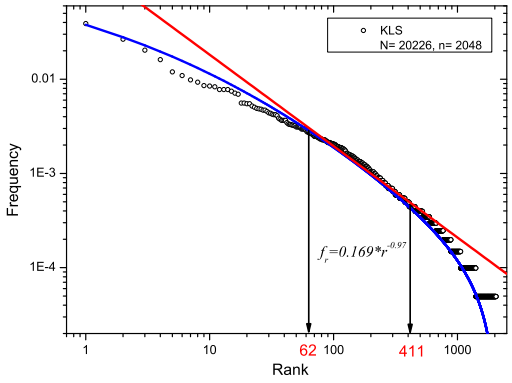


FIG. 1: (Color online) Frequency versus rank for the short modern Chinese text KLS; see Appendix C for its description. Red line: the Zipf curve $f_r = 0.169r^{-0.97}$; see Table I. Arrows and red numbers indicate on the validity range of the Zipf's law. Blue line: the numerical solution of (17, 18) for $c = 0.169$. It coincides with the generalized Zipf law (21) for $r > r_{\rm min} = 62$. The step-wise behavior of f_r for $r > r_{\rm max}$ refers to hapax legomena.

As a measure of fitting quality one can take:

$$\min_{c,\gamma}[SS_{\rm err}(c,\gamma)] = SS_{\rm err}(c^*,\gamma^*) = SS^*_{\rm err}. \tag{7}$$

This is however not the only relevant quality measure. Another (more global) aspect of this quality is the coefficient of correlation between $\{y_r\}_{r=1}^n$ and $\{\hat{y}_r\}_{r=1}^n$ [2, 48]

$$R^2 = \frac{\left[\sum_{k=1}^n (y_k - \overline{y})(\hat{y}^*_k - \overline{\hat{y}^*})\right]^2}{\sum_{k=1}^n (y_k - \overline{y})^2 \sum_{k=1}^n (\hat{y}^*_k - \overline{\hat{y}^*})^2}, \tag{8}$$

where

$$\hat{y}^* = \{\hat{y}^*_r = \ln c^* - \gamma^* x_r\}_{r=1}^n, \quad \overline{\hat{y}^*} \equiv \frac{1}{n}\sum\nolimits_{k=1}^n \hat{y}^*_k. \tag{9}$$

For the linear fitting (5) the squared correlation coefficient is equal to the coefficient of determination,

$$R^2 = \sum\nolimits_{k=1}^n (\hat{y}^*_k - \overline{y})^2 \Big/ \sum\nolimits_{k=1}^n (y_k - \overline{y})^2, \tag{10}$$

the amount of variation in the data explained by the fitting [2, 48]. Hence $SS^*_{\rm err} \to 0$ and $R^2 \to 1$ mean good fitting. We minimize $SS_{\rm err}$ over c and γ for $r_{\rm min} \leq r \leq r_{\rm max}$ and find the maximal value of $r_{\rm max} - r_{\rm min}$ for which $SS^*_{\rm err}$ and $1 - R^2$ are smaller than, respectively, 0.05 and 0.005. This value of $r_{\rm max} - r_{\rm min}$ also determines the final fitted values c^* and γ^* of c and γ, respectively; see Tables I, II, III and Fig. 1. Thus c^* and γ^* are found simultaneously with the validity range $[r_{\rm max}, r_{\rm max}]$ of the law. Whenever there is no risk of confusion, we for simplicity refer to c^* and γ^* as c and γ, respectively.

B. Empiric results on the Zipf's law

Here are results produced via the above linear fitting.

TABLE I: Parameters of the modern Chinese texts (see Appendix C for further details). N is the total number of characters in the text. The number of different characters is n. The Zipf's law $f_r = cr^{-\gamma}$ holds for the ranks $r_{\min} \leq r \leq r_{\max}$; see section III A. Here $\sum_{k<r_{\min}} f_k$ and $\sum_{k=r_{\min}}^{r_{\max}} f_k$ are the total frequencies carried out by the pre-Zipfian and Zipfian domain, respectively. d is the difference between the total frequency of the Zipfian domain got empirically and its value according to the Zipf's law: $d = \sum_{k=r_{\min}}^{r_{\max}} (ck^{-\gamma} - f_k)$. Its absolute value d characterizes the global precision of the Zipf's law.
AQZ & KLS means joining the texts AQZ and KLS.
r_b is the conventional borderline rank between the exponential-like range and the hapax legomena; see section IV B 2 for its definition. Whenever we put "-" instead of it, we mean that either the exponential-like range is absent or it is not distinguishable from the hapax legomena.

Texts	N	n	$r_{\min}$	$r_{\max}$	c	γ	$\sum_{k<r_{\min}} f_k$	$\sum_{k=r_{\min}}^{r_{\max}} f_k$	$\lvert d\rvert$	r_b
AQZ	18153	1553	56	395	0.2239	1.03	0.42926	0.38424	0.00624	-
KLS	20226	2047	62	411	0.169	0.97	0.39971	0.379728	0.005728	-
AQZ & KLS	38379	2408	66	439	0.195	1.0	0.41684	0.369	0.0022	-
PFSJ	705130	3820	67	583	0.234	1.03	0.39544	0.425379	0.00842	1437
SHZ	704936	4376	78	590	0.225	1.02	0.39905	0.42	0.009561	1618

TABLE II: Parameters of classic Chinese texts (see Appendix C for further details). Notations have the same meaning as in Table I. Here 4 texts means joining of the texts CQF, SBZ, WJZ, and HLJ. Also, 7 (10,14) texts mean joining of the 4 with other 3 (6,11) classic texts, which we do not mention separately, because they give no new information.

Texts	N	n	$r_{\min}$	$r_{\max}$	c	γ	$\sum_{k<r_{\min}} f_k$	$\sum_{k=r_{\min}}^{r_{max}} f_k$	$\lvert d\rvert$	r_b
CQF	30017	1661	47	365	0.1778	0.985	0.43906	0.38997	0.00441	-
SBZ	24634	1959	52	357	0.1819	0.972	0.42828	0.408787	0.004353	-
WJZ	26330	1708	46	360	0.208	0.999	0.40434	0.418733	0.006923	-
HLJ	26559	1837	56	372	0.209	1.01	0.43674	0.379454	0.000832	-
CQF & SBZ	54651	2528	68	483	0.19498	0.989	0.42031	0.401661	0.00483	-
CQF & WJZ	56347	2302	66	439	0.20654	1.002	0.42815	0.383514	0.00564	-
CQF & HLJ	56576	2458	65	416	0.19498	0.998	0.43138	0.38654	0.00913	-
SBZ & WJZ	50964	2505	68	465	0.20512	0.992	0.40116	0.409017	0.00382	-
SBZ & HLJ	51193	2608	72	423	0.20893	1.000	0.41157	0.369598	0.00798	-
WJZ & HLJ	52889	2303	66	432	0.23988	1.035	0.43044	0.380801	0.002321	-
4 texts	107540	3186	75	528	0.22387	1.021	0.42526	0.391818	0.0007	681
7 texts	190803	4069	57	513	0.158	0.97	0.39381	0.4102	0.00331	789
10 texts	278557	4727	67	552	0.168	0.978	0.38058	0.4015	0.00217	1015
14 texts	348793	5018	78	625	0.176	0.98	0.39116	0.418983	0.00954	1223
SJ	572864	4932	76	535	0.236	1.025	0.40153	0.41253	0.007564	1336

1. For each Chinese text there is a specific (Zipfian) range of ranks $r \in [r_{\min}, r_{\max}]$, where the Zipf's law $f_r = cr^{-\gamma}$ holds with $\gamma \approx 1$ and $c \lesssim 0.25$; see Tables I, II and Fig. 1. Both for $r < r_{\min}$ and $r > r_{\max}$ the frequencies are below the Zipf curve; see Fig. 1. A power rank-frequency relation with exponent $\gamma \approx 1$ is the hallmark of the Zipf's law [1–4].

Note that though the validity range $|r_{\max} - r_{\min}|$ is few times smaller than the maximal rank n (see Tables I and II and Figs. 1 and 2), it is relevant, since it contains a sizable amount of the overall frequency: for Chinese texts (short or long) the Zipfian range carries 40 % of the overall frequency, i.e. $\sum_{k=r_{\min}}^{r_{\max}} f_k \simeq 0.4$.

2. In the pre-Zipfian range $1 \leq r < r_{\min}$ the overall number of function and empty characters is more than the number of content characters. Function and empty characters serve for establishing grammatical constructions (e.g. "的" *(de)*, "是" *(shì)*, "了" *(le)*, "不" *(bù)*, "在" *(zài)*). (We shall list them separately, though for our purposes they can be joined together; the main difference between them is that the empty characters are not used alone.)

But the majority of characters in the Zipfian range do have a specific meaning (content characters). A subset of those content characters has a meaning that is specific for the text and can serve as its key-characters; see Appendix D and Table IX for an example.

Let us take for an example the modern Chinese text KLS; see Table I (this text concerns military activities; see Appendix C). The pre-Zipfian range of this text con-

TABLE III: Parameters of four English texts and their mixtures: *The Age of Reason* (AR) by T. Paine, 1794 (the major source of British deism). *Time Machine* (TM) by H. G. Wells, 1895 (a science fiction classics). *Thoughts on the Funding System and its Effects* (TF) by P. Ravenstone, 1824 (economics). *Dream Lover* (DL) by J. MacIntyre, 1987 (a romance novella). TF & TM means joining the texts TF and TM.
The total number of words N, the number of different words n, the lower $r_{\min}$ and the upper $r_{\max}$ ranks of the Zipfian domain, the fitted values of c and γ, the overall frequencies of the pre-Zipfian and Zipfian range, and the difference d between the total frequency of the Zipfian domain got empirically and its value according to the Zipf's law: $d = \sum_{k=r_{\min}}^{r_{\max}} (ck^{-\gamma} - f_k)$.

Texts	N	n	$r_{\min}$	$r_{\max}$	c	γ	$\sum_{k<r_{\min}} f_k$	$\sum_{k=r_{\min}}^{r_{\max}} f_k$	$\lvert d\rvert$
TF	26624	2067	36	371	0.168	1.032	0.44439	0.35158	0.00333
TM	31567	2612	42	332	0.166	1.041	0.45311	0.33876	0.01004
AR	22641	1706	32	339	0.178	1.038	0.47254	0.33947	0.00048
DL	24990	1748	34	230	0.192	1.039	0.47955	0.33251	0.02145
TF & TM	54191	3408	30	602	0.139	1.013	0.43508	0.40876	0.02091
TF & AR	45265	2656	33	628	0.138	0.998	0.45468	0.41045	0.00239
TF & DL	47614	2877	28	527	0.162	1.014	0.42599	0.42261	0.01490
TM & AR	54208	3184	43	592	0.157	1.021	0.47582	0.39687	0.00491
TM & DL	56557	3154	45	493	0.161	1.023	0.46726	0.38456	0.01211
AR & DL	47631	2550	38	496	0.165	1.012	0.45375	0.39236	0.00947
Four texts	101822	4047	39	927	0.158	1.015	0.44245	0.44158	0.00187

tains 61 characters. Among them there are, 24 function characters, 9 empty characters, 25 content characters, and finally there are 3 key-characters [15]: horn "号" *(hào)*, army "军" *(jūn)* and soldier "兵" *(bīn)*.

The Zipfian range of the KLS contains 350 characters. Among them, 91 are function, 10 are empty, 230 are content and 19 are key-characters (see Appendix D for the full list of key-characters for this text).

3. The absolute majority of different characters with ranks in $[r_{\min}, r_{\max}]$ have different frequencies. Only for $r \simeq r_{\max}$ the number of different characters having the same frequency is $\simeq 10$. For $r > r_{\max}$ we meet the hapax legomena effect: characters occurring only few times in the text (i.e. $f_r N = 1, 2, 3...$ is a small integer), and many characters having the same frequency f_r [3]. The effect is not described by any smooth rank-frequency relation, including the Zipf's law. Hence for short texts we get that the Zipf's law holds for as high ranks as possible, in the sense that for $r > r_{\max}$ no smooth rank-frequency relations are possible at all.

Note that the very existence of hapax legomena is a non-trivial effect, since one can easily imagine (artificial) texts, where (say) no character appear only once. The theory reviewed below allows to explain the hapax legomena range together with the Zipf's law; see below. It also predicts a generalization of the Zipf's law to frequencies $r < r_{\min}$ that is more adequate (than the Zipf's law) to the empiric data; see Figs. 1 and 2.

4. All the above results hold for relatively short English [17]; see Table III and Fig. 2. In particular, the Zipfian range of English texts also contains mainly content words including the keywords. This is known and is routinely used in document processing [33].

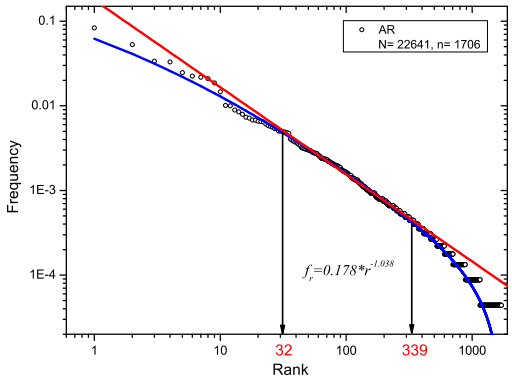


FIG. 2: (Color online) Frequency vs. rank for the English text AR; see Table III. Red line: the Zipf curve $f_r = 0.178 r^{-1.038}$. Other notations have the same meanings as in Fig. 1.

We thus conclude that as far as short texts are concerned, the Zipf's law holds for Chinese characters in the same way as it does for English words.

5. To check our results on fitting the empiric data for word frequencies to the Zipf's law we carried out three alternative tests.

5.1 First we applied the Kolmogorov-Smirnov (KS) test to decide on the fitting quality of the data with the Zipf's law (in the range $[r_{\min}, r_{\max}]$). The test was carried out both with and without transforming to the logarithmic coordinates (3) and it fully confirmed our result;

[15] We present that meaning of the character which is most relevant in the context of the text.

see Table IV. For a detailed presentation of the KS test results see Appendix E and Table X therein.

5.2 It was recently shown that even when the applicability range $[r_{\min}, r_{\max}]$ of a power law is known, the linear least-square method (that we employed above) may not give accurate estimations for the exponent γ of the power law [49–51]. It was then argued that the method of Maximum Likelihood Estimation (MLE) is more reliable in this context. Hence to show that our results are robust, we calculated γ using the MLE method, as suggested in [49–51]. We got that the difference with the linear least square method is quite small (changes come only at the third decimal place); see Table IV.

5.3 We also checked whether our results on the power law exponent γ are stable with respect to non-linear fitting schemes, the ones that do not employ the logarithmic coordinates (3), but operate directly with the form (2). Again, we find that non-linear fitting schemes (that we carried out via routines of Mathematica 7) produce very similar results for γ; see Table IV.

One reason for such a good coincidence between our linear fitting results and alternative tests is that we use a rather strict criteria ($SS^*_{\rm err} < 0.05$ and $R^2 > 0.995$) for determining *first* the Zipfian range $[r_{\min}, r_{\max}]$ and then the parameters of the Zipf's law. Another reason is that in the vicinity of $r_{\max}$, the number of different words having the same frequency is not large (it is smaller than 10). Hence there are no problems with lack of data points or systematic biases that can plague the applicability of the least square method for determination of the exponent γ.

C. Theoretical description of the Zipf's law and hapax legomena

1. Assumptions of the model

A theoretical description of the Zipf's law that is specifically applicable to short English texts was recently proposed in [17]; it is reviewed below. The theory is based on the ideas of latent semantic analysis and the concept of mental lexicon [17]. We shall now briefly remind it to demonstrate that

– The rank-frequency relation for short Chinese and English texts can be described by the same theory.

– The theory allows to extrapolate the Zipf's law to high and low frequencies (including hapax legomena).

– It allows to understand the bound $c < 0.25$ for the prefactor of the Zipf's law (since the law does not apply for all frequencies, c is not fixed from normalization).

– The theory confirms the intuitive expectation about the difference between the Zipfian and hapax legomena range: in the first case the probability of a word is equal to its frequency (frequent words). In the hapax legomena range, both the probability and frequency are small and different from each other.

– In the following section the theory is employed for describing the rank-frequency relation of Chinese characters outside of the validity range of the Zipf's law.

Our model for deriving the Zipf's law together with the description of the hapax legomena makes four assumptions (see [17] for further details). Below we shall refer to the units of the text as words; whenever this theory applies for Chinese texts we shall mean characters instead of words.

• The *bag-of-words picture* focusses on the frequency of the words that occur in a text and neglects their mutual disposition (i.e. syntactic structure) [52]. This is a natural assumption for a theory describing word frequencies, which are invariant with respect to an arbitrary permutation of the words in a text. The latter point was recently verified in [53].

Given n different words $\{w_k\}_{k=1}^n$, the joint probability for w_k to occur $\nu_k \geq 0$ times in a text T is assumed to be multinomial

$$\pi[\boldsymbol{\nu}|\boldsymbol{\theta}] = \frac{N!\,\theta_1^{\nu_1}...\theta_n^{\nu_n}}{\nu_1!...\nu_n!}, \quad \boldsymbol{\nu} = \{\nu_k\}_{k=1}^n, \quad \boldsymbol{\theta} = \{\theta_k\}_{k=1}^n \tag{11}$$

where $N = \sum_{k=1}^n \nu_k$ is the length of the text (overall number of words), ν_k is the number of occurrences of w_k, and θ_k is the probability of w_k.

Hence according to (11) the text is regarded to be a sample of word realizations drawn independently with probabilities θ_k.

The bag-of-words picture is well-known in computational linguistics [52]. But for our purposes it is incomplete, because it implies that each word has the same probability for different texts. In contrast, it is well known (and routinely confirmed by the rank-frequency analysis) that the same words do *not* occur with same frequencies in different texts.

• To improve this point we make $\boldsymbol{\theta}$ a random vector with a text-dependent density $P(\boldsymbol{\theta}|T)$ (a similar, but stronger assumption was done in [52]). With this assumption the variation of the word frequencies from one text to another will be explained by the randomness of the word probabilities.

We now have three random objects: text T, probabilities $\boldsymbol{\theta}$ and the occurrence numbers $\boldsymbol{\nu}$. Since $\boldsymbol{\theta}$ was introduced to explain the relation of T with $\boldsymbol{\nu}$, it is natural to assume that the triple $(T, \boldsymbol{\theta}, \boldsymbol{\nu})$ form a Markov chain: the text T influences the observed $\boldsymbol{\nu}$ only via $\boldsymbol{\theta}$. Then the probability $p(\boldsymbol{\nu}|T)$ of $\boldsymbol{\nu}$ in a given text T reads

$$p(\boldsymbol{\nu}|T) = \int \mathrm{d}\boldsymbol{\theta}\, \pi[\boldsymbol{\nu}|\boldsymbol{\theta}]\, P(\boldsymbol{\theta}|T). \tag{12}$$

This form of $p(\boldsymbol{\nu}|T)$ is basic for probabilistic latent semantic analysis [54], a successful method of computational linguistics. There the density $P(\boldsymbol{\theta}|T)$ of latent variables $\boldsymbol{\theta}$ is determined from the data fitting. We shall deduce $P(\boldsymbol{\theta}|T)$ theoretically.

• The text-conditioned density $P(\boldsymbol{\theta}|T)$ is generated from a prior density $P(\boldsymbol{\theta})$ via conditioning on the or-

TABLE IV: Comparison between different methods of estimating the exponent γ of the Zipf's law; see (1): LLS (linear least-square), NLS (nonlinear least-square), MLE (maximum likelihood estimation). We also present the p-value of the KS test when comparing the empiric word frequencies in the range $[r_{\min}, r_{\max}]$ with the Zipf's-law within the linear lest-square method (LLS); for a more detailed presentation of the KS results see Appendix E. Recall that the p-values have to be sufficiently larger than 0.1 for fitting to be reliable from the viewpoint of KS test. This holds for the presented data; see Appendix E for details.

Texts	γ, LLS	γ, NLS	γ, MLE	p-value
TF	1.032	1.033	1.035	0.865
TM	1.041	1.036	1.039	0.682
AR	1.038	1.042	1.044	0.624
DL	1.039	1.034	1.035	0.812
AQZ	1.03	1.028	1.027	0.587
KLS	0.97	0.975	0.973	0.578
CQF	0.985	0.983	0.981	0.962
SBZ	0.972	0.967	0.973	0.796
WJZ	0.999	0.993	0.995	0.852
HLJ	1.01	1.015	1.011	0.923

dering of $\mathbf{w} = \{w_k\}_{k=1}^n$ in T:

$$P(\boldsymbol{\theta}|T) = P(\boldsymbol{\theta})\,\chi_T(\boldsymbol{\theta},\mathbf{w}) \Big/ \int \mathrm{d}\boldsymbol{\theta}'\, P(\boldsymbol{\theta}')\,\chi_T(\boldsymbol{\theta}',\mathbf{w})\,. \quad (13)$$

Thus if different words of T are ordered as $(w_1, ..., w_n)$ with respect to the decreasing frequency of their occurrence in T (i.e. w_1 is more frequent than w_2), then $\chi_T(\boldsymbol{\theta},\mathbf{w}) = 1$ if $\theta_1 \geq ... \geq \theta_n$, and $\chi_T(\boldsymbol{\theta},\mathbf{w}) = 0$ otherwise.

• The apriori density of the word probabilities $P(\boldsymbol{\theta})$ in (13) can be related to the mental lexicon (store of words) of the author prior to generating a concrete text. For simplicity, we assume that the probabilities θ_k are distributed identically [see [17] for a verification of this assumption] and the dependence among them is due to $\sum_{k=1}^n \theta_k = 1$ only:

$$P(\boldsymbol{\theta}) \propto u(\theta_1)\,...\,u(\theta_n)\,\delta(\sum\nolimits_{k=1}^{n}\theta_k - 1), \quad (14)$$

where $\delta(x)$ is the delta function and the normalization ensuring $\int_0^\infty \prod_{k=1}^n \mathrm{d}\theta_k\, P(\boldsymbol{\theta}) = 1$ is omitted.

2. *Zipf's law*

It remains to specify the function $u(\theta)$ in (14). Ref. [17] reviews in detail the experimentally established features of the human mental lexicon (see [55] in this context) and deduces from them that the suitable function $u(\theta)$ is

$$u(f) = (n^{-1}c + f)^{-2}, \quad (15)$$

where c is to be related to the prefactor of the Zipf's law.

Above equations (11–15) together with the feature $n^3 \gg N \gg 1$ of real texts (where n is the number of different words, while N is the total number of words in the text) allow to the final outcome of the theory: the probability $p_r(\nu|T)$ of the character (or word) with the rank r to appear ν times in a text T (with N total characters and n different characters) [17]:

$$p_r(\nu|T) = \frac{N!}{\nu!(N-\nu)!}\phi_r^\nu(1-\phi_r)^{N-\nu}, \quad (16)$$

where the effective probability ϕ_r of the character is found from two equations for two unknowns μ and ϕ_r:

$$r/n = \int_{\phi_r}^\infty \mathrm{d}\theta\,\frac{e^{-\mu\theta}}{(c+\theta)^2} \Big/ \int_0^\infty \mathrm{d}\theta\,\frac{e^{-\mu\theta}}{(c+\theta)^2}\,, \quad (17)$$

$$\int_0^\infty \mathrm{d}\theta\,\frac{\theta\,e^{-\mu\theta}}{(c+\theta)^2} = \int_0^\infty \mathrm{d}\theta\,\frac{e^{-\mu\theta}}{(c+\theta)^2}, \quad (18)$$

where c is a constant that will later on shown to coincide with the prefactor of the Zipf's law.

For $c \lesssim 0.25$, $c\mu$ determined from (18) is small and is found from integration by parts:

$$\mu \simeq c^{-1}\,e^{-\gamma_{\mathrm{E}} - \frac{1+c}{c}}, \quad (19)$$

where $\gamma_{\mathrm{E}} = 0.55117$ is the Euler's constant. One solves (17) for $c\mu \to 0$:

$$\frac{r}{n} = ce^{-n\phi_r\mu}/(c + n\phi_r). \quad (20)$$

Recall that according to (16), ϕ_r is the probability for the character (or the word in the English situation) with rank r. If ϕ_r is sufficiently large, $\phi_r N \gg 1$, the character with rank r appears in the text many times and its frequency $\nu \equiv f_r N$ is close to its maximally probable value $\phi_r N$; see (16). Hence the frequency f_r can be obtained via the probability ϕ_r. This is the case in the Zipfian domain, since according to our empirical results (both for

Chinese and English) $\frac{1}{n} \lesssim f_r$ for $r \leq r_{\max}$, and—upon identifying $\phi_r = f_r$—the above condition $\phi_r N \gg 1$ is ensured by $N/n \gg 1$; see Tables I, II and III.

Let us return to (20). For $r > r_{\min}$, $\phi_r n\mu = f_r n\mu < 0.04 \ll 1$; see (19) and Figs. 1 and 2. We get from (20):

$$f_r = c(r^{-1} - n^{-1}). \tag{21}$$

This is the Zipf's law generalized by the factor n^{-1} at high ranks r. This cut-off factor ensures faster [than r^{-1}] decay of f_r for large r.

Figs. 1 and 2 shows that (21) reproduces well the empirical behavior of f_r for $r > r_{\min}$. Our derivation shows that c is the prefactor of the Zipf's law, and that our assumption on $c \lesssim 0.25$ above (19) agrees with observations; see Tables I, II and III.

For given prefactor c and the number of different characters n, (17) predict the Zipfian range $[r_{\min}, r_{\max}]$ in agreement with empirical results; see Figs. 1 and 2.

For $r < r_{\min}$, it is not anymore true that $f_r n\mu \ll 1$ (though it is still true that $f_r N = \phi_r N \gg 1$). So the fuller expression (17) is to be used instead of (20). It reproduces qualitatively the empiric behavior of f_r also for $r < r_{\min}$; see Figs. 1 and 2. We do not expect any better agreement theory and observations for $r < r_{\min}$, since the behavior of frequencies in this range is irregular and changes significantly from one text to another.

D. Hapax legomena

1. *Hapax legomena as a consequence of the generalized Zipf's law*

According to (16), the probability ϕ_r is small for $r \gg r_{\max}$ and hence the occurrence number $\nu \equiv f_r N$ of the character with the rank r is a small integer (e.g. 1 or 2) that cannot be approximated by a continuous function of r; see Figs. 1 and 2. In particular, the reasoning after (20) on the equality between frequency and probability does not apply, although we see in Figs. 1 and 2 that (21) roughly reproduces the trend of f_r even for $r > r_{\max}$.

To describe this hapax legomena range, define r_k as the rank, when $\nu \equiv f_r N$ jumps from integer k to $k+1$ (hence the number of characters that appear $k+1$ times is $r_k - r_{k+1}$). Since ϕ_r reproduces well the trend of f_r even for $r > r_{\max}$, see Fig. 1, r_k can be theoretically predicted from (21) by equating its left-hand-side to k/N:

$$\hat{r}_k = [\frac{k}{Nc} + \frac{1}{n}]^{-1}, \qquad k = 0, 1, 2, ... \tag{22}$$

Eq. (22) is exact for $k = 0$, and agrees with r_k for $k \geq 1$; see Table V. We see that a single formalism describes both the Zipf's law for short texts and the hapax legomena range. We stress that for describing the hapax legomena no new parameters are needed; it is based on the same parameters N, n, c that appear in the Zipf's law.

2. *Comparing with previous theories of hapax legomena*

Several theories were proposed over the years for describing the hapax legomena range; see [56] for a review. To be precise, these theories were proposed for rare words (not for rare Chinese characters), but since the Zipf's law applies to characters, we expect that these theories will be relevant. We now compare predictions of the main theories with (22). The latter turns out to be superior.

Recall that for obtaining (22) it is necessary to employ the generalized (by the factor n^{-1}) form (21) of the Zipf's law. The correction factor is not essential in the proper Zipfian domain (since it is a pure power law), but is crucial for obtaining a good agreement with empiric data in the hapax legomena range; see Figs. 1 and 2. The influence of this correcting factor can be neglected for $k \gg Nc/n$ in (22), where we get

$$\hat{r}_{k-1} - \hat{r}_k \propto \frac{1}{k(k-1)}, \tag{23}$$

for the number of characters having frequency k/N. This relation, which is a crude particular case of (22), is sometimes called the second Zipf's law, or the Lotka's law [3, 56]. The applicability of (23) is however limited, e.g. it does not apply to the data shown in Table V.

Another approach to frequencies of rare words was proposed in [57]; see [56] for a review. Its basic result (24) was recently recovered from a partial maximization of entropy (random group formation approach) [58] [16]. It makes the following prediction for the number $nP(k)$ [17] of characters that appear in the text k times (i.e. $P(k)$ is a prediction for $(r_{k-1} - r_k)/n$)

$$P(k) \propto e^{-bk}\, k^{-\gamma}, \qquad 1 \leq k \leq f_1 N, \tag{24}$$

where we omitted the normalization ensuring $\sum_{k=1}^{f_1 N} P(k) = 1$, and where the constants $b > 0$ and $\gamma > 0$ are determined from three parameters of the text: the overall number of characters N, the number of different characters n and the maximal frequency f_1 [58]. Distributions similar to (24) (i.e. exponentially modified power-laws) were derived from partial maximization of

[16] Ref. [58] presented a broad range of applications, but it did not study Chinese characters. We acknowledge one of the referees of this work who informed us that such unpublished studies do exist: Chinese characters are within the applicability range of Ref. [58], as we confirm in Table V. The predictions of (24) for the AQZ text that we reproduce in Table V were communicated to us by the referee.

[17] Please do not mix up $P(k)$ with the density of character probabilities that appear in (14, 15). Indeed, $P(k)$ is defined as empiric frequency; it has a discrete argument and applies to any collection of objects, also the one that was generated by any probabilistic mechanism. In contrast, (14, 15) amount to a density of probabities that has continuous argument(s) and assumes a specific generative model.

entropy prior to Ref. [58] (e.g. in [59, 60]), but it was Ref. [58] that emphasized their broad applicability.

Note that $P(k)$ in (24) does not apply out of the hapax legomena range, where for all k we must have $P(k) = 1/n$. However, it is expected that for $n \gg 1$ this discrepancy will not hinder the applicability of (24) to $P(k)$ with sufficiently small values of k, i.e. within the hapax legomena range.

The results predicted by (24) are compared with our data in Table V. For clarity, we transform (24) to a prediction $\widetilde{r}_k$ for quantities r_k:

$$\widetilde{r}_l = n[1 - \sum_{k=1}^{l} P(k)], \quad l \geq 1, \tag{25}$$

i.e. we go to the cumulative distribution function $\sum_{k=1}^{l} P(k)$.

While the predictions of (25) are in a certain agreement with the data, their accuracy is inferior (at least by an order of magnitude) as compared to predictions of (22); see Table V. The reason of this inferiority is that though both (24) and (22) use three input parameters, (24) is not sufficiently specific to the studied text.

Finally, let us turn to the Waring-Herdan approach which predicts for $nP(k)$ (the number of characters that appear in the text k times) a version of the Yule's distribution [56]:

$$P(k+1) = P(k)\,\frac{a+k-1}{x+k}, \quad k \geq 1, \tag{26}$$

where a and x are expressed via three (the same number as in the previous two approaches) input parameters N (the overall number of characters), n (the number of distinct characters) and $nP(1)$ (the number of characters that appear only once) [56]:

$$a = \left(\frac{1}{1-P(1)} - P(1) - 1\right)^{-1}, \quad x = \frac{a}{1-P(1)}. \tag{27}$$

Eqs. (26, 27) are turned to a prediction r'_k for r_k. As Table VI shows, these predictions [18] are also inferior as compared to those of (22), especially for $k \geq 5$.

E. Summary

It is to be concluded from this section that—as far as the applicability of the Zipf's law to short texts is concerned—the Chinese characters behave similarly to English words. In particular, both situations can be adequately described by the same theory. In particular, the hapax legomena range of short texts is described via the generalized Zipf's law.

We should like to stress again why the consideration of short texts is important. One can argue that—at least for the sake of rank-frequency relations—long texts are just mixtures (joinings) of shorter, thematically homogeneous pieces (this premise is fully confirmed below). Hence the task of studying rank-frequency relations separates into two parts: first understanding short texts, and then long ones. We now move to the second part.

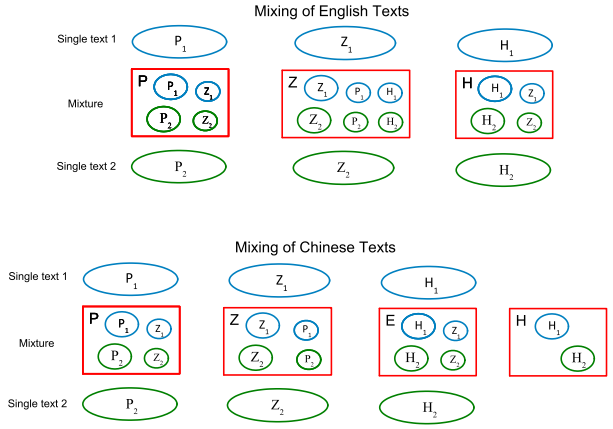


FIG. 3: Schematic representation of various ranges under mixing (joining) two English (upper figure) and two Chinese (lower figure) texts. P_k, Z_k and H_k mean, respectively, the pre-Zipfian, Zipfian and hapax legomena ranges of the text k ($k = 1, 2$). P, Z and H mean the corresponding ranges for the mixture of texts 1 and 2. E means the exponential-like range that emerges upon mixing of two Chinese texts. For each range of the mixture we show schematically contributions from various ranges of the separate texts. The relative importance of each contribution is conventionally represented by different magnitudes of the circles.

IV. RANK-FREQUENCY RELATION FOR LONG TEXTS AND MIXTURES OF TEXTS

A. Mixing English texts

When mixing (joining)[19] different English texts the validity range of the Zipf's law increases due to acquiring more higher rank words, i.e. $r_{\min}$ stays approximately fixed, while $r_{\max}$ increases; see Table III. The overall

[18] Eq. (26) can viewed as a consequence of the Simon's model of text generation. This model does not apply to real texts as was recently demonstrated in [53]. Nevertheless (26) keeps its relevance as a convenient fitting expression; see also [56] in this context.

[19] Upon joining two texts (A and B), the word frequencies get mixed: $f_k(\mathrm{A\&B}) = \frac{N_\mathrm{A}}{N_\mathrm{A}+N_\mathrm{B}} f_k(\mathrm{A}) + \frac{N_\mathrm{B}}{N_\mathrm{A}+N_\mathrm{B}} f_k(\mathrm{B})$, where N_A and $f_k(\mathrm{A})$ are, respectively, the total number of words and the frequency of word k in the text A.

TABLE V: The hapax legomena range for Chinese characters demonstrated for 4 short Chinese texts. The first and second text are in Modern Chinese, other two are in Classic Chinese; see Tables I and II. r_k is defined before (22) and is found from empirical data, while $\hat{r}_k$ is calculated from (22); see section III D. We also present the relative error for $\hat{r}_k$ approximating r_k.

Texts	k	1	2	3	4	5	6	7	8	9	10
AQZ	r_k	1097	857	702	595	522	461	414	370	339	311
	$\hat{r}_k$	1116	869	711	601	520	458	409	369	336	308
	$\frac{\lvert\hat{r}_k - r_k\rvert}{r_k}$	0.017	0.014	0.013	0.010	0.0038	0.0065	0.012	0.0027	0.0088	0.0096
KLS	r_k	1405	1060	885	767	662	582	520	455	408	377
	$\hat{r}_k$	1428	1093	884	750	656	575	515	445	404	369
	$\frac{\lvert\hat{r}_k - r_k\rvert}{r_k}$	0.016	0.031	0.0011	0.022	0.0091	0.012	0.0096	0.022	0.0098	0.021
SBZ	r_k	1460	1141	959	850	735	676	618	563	517	481
	$\hat{r}_k$	1481	1168	980	848	740	656	599	553	497	488
	$\frac{\lvert\hat{r}_k - r_k\rvert}{r_k}$	0.014	0.024	0.022	0.0024	0.0068	0.029	0.031	0.018	0.039	0.015
HLJ	r_k	1302	1045	872	756	669	604	551	501	467	430
	$\hat{r}_k$	1327	1080	900	783	684	607	545	494	462	420
	$\frac{\lvert\hat{r}_k - r_k\rvert}{r_k}$	0.019	0.033	0.032	0.035	0.022	0.0049	0.011	0.014	0.011	0.023

TABLE VI: The hapax legomena range for 2 Chinese texts; see Table I and cf. with Table V. We compare the relative errors for, respectively, $\hat{r}_k$ (given by (22)) $\widetilde{r}_k$ and r'_k in approximating the data r_k; see section III D 2. Here $\widetilde{r}_k$ is defined by (25, 24), and r'_k is the prediction made by (26, 27). For AQZ the parameters in (24) are $\gamma = 1.443$ and $b = 0.0049$. For KLS: $\gamma = 1.574$ and $b = 0.0033$. It is seen that the relative error provided by $\hat{r}_k$ is always smaller; the only exclusion is the case $k = 2$ of the KLS text. Recall that $r'_1 = r_1$ by definition.

Texts	k	1	2	3	4	5	6	7	8	9	10
AQZ	$\lvert\widetilde{r}_k - r_k\rvert / r_k$	0.141	0.161	0.153	0.138	0.129	0.112	0.097	0.069	0.057	0.039
	$\lvert r'_k - r_k\rvert / r_k$	0	0.025	0.048	0.071	0.102	0.121	0.141	0.146	0.163	0.174
	$\lvert\hat{r}_k - r_k\rvert / r_k$	0.017	0.014	0.013	0.010	0.0038	0.0065	0.012	0.0027	0.0088	0.0096
KLS	$\lvert\widetilde{r}_k - r_k\rvert / r_k$	0.194	0.221	0.245	0.267	0.259	0.250	0.240	0.206	0.183	0.179
	$\lvert r'_k - r_k\rvert / r_k$	0	0.0087	0.063	0.114	0.137	0.157	0.176	0.168	0.170	0.190
	$\lvert\hat{r}_k - r_k\rvert / r_k$	0.016	0.031	0.0011	0.022	0.0091	0.012	0.0096	0.022	0.0098	0.021

precision of the Zipf's law also increases upon mixing, as Table III shows.

The rough picture of the evolution of the rank-frequency relation under mixing two texts is summarized as follows; see Table III and Fig. 3 for a schematic illustration. The majority of the words in the Zipfian range of the mixture (e.g. AR & TM) come from the Zipfian ranges of the separate texts. In particular, all the words that appear in the Zipfian ranges of the separate words do appear as well in the Zipfian range of the mixture (e.g. the Zipfian ranges of AR and TM have 130 common words). There are also relatively smaller contributions to the Zipfian range of the mixture from the pre-Zipfian and hapax legomena range of separate texts: note from Table III that the Zipfian range of the mixture AR & TM is 82 words larger than the sum of two separate Zipfian ranges, which is (307 + 290) minus 130 common words.

Some of the words that appear only in the Zipfian range of one of separate texts will appear in the hapax legomena range of the mixture; other words move from the pre-Zipfian range of separate texts to the Zipfian range of the mixture. But these are relatively minor effects: the rough effect of mixing is visualized by saying that the Zipfian ranges of both texts combine to become a larger Zipfian range of the mixture and acquire additional words from other ranges of the separate texts; see Fig. 3. Note that the keywords of separate words stay in the Zipfian range of the mixture, e.g. after joining all four above texts, the keywords of each text are still in the Zipfian range (which now contains almost 900 words); see Table III.

The results on the behavior of the Zipf's law under mixing are new, but their overall message—the validity of the Zipf's law improves upon mixing—is expected, since it is known that the Zipf's law holds not only for short but also for long English texts and for frequency dictionaries (huge mixtures of various texts) [1–4].

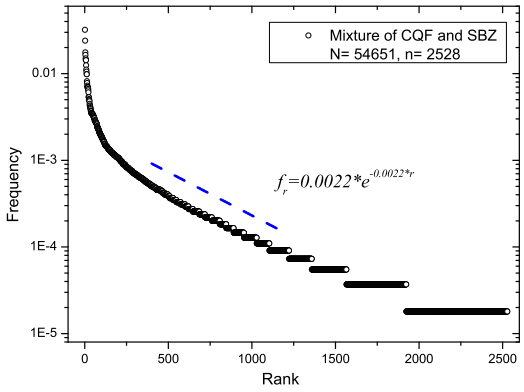


FIG. 4: (Color online) Rank frequency distribution for the mixture of CQF and SBZ.; see Tables I and II and Appendix C. The scale of the frequency is chosen such that the exponential-like range of the rank-frequency relation for $r > 500$ is made visible. For comparison, the dashed blue line shows a curve $f_r = 0.0022e^{-0.0022r}$. For the present example, the exponential-like range is essentially mixed with hapax legomena, since for frequencies f_r with $r > r_{\max}$ the number of different words having this frequency is larger than 10. Recall that the Zipf's law holds for $r_{\min} < r < r_{\max}$; see Tables I and II.

B. Mixing Chinese texts

1. *Stability of the Zipfian range*

The situation for Chinese texts is different. Upon mixing two Chinese texts the validity range of the Zipf's law increases, but much slower as compared to English texts; see Tables I and II. The validity ranges of the separate texts do not combine (in the above sense of English texts). Though the common words in the Zipfian ranges of separate texts do appear in the Zipfian range of the mixture, a sizable amount of those words that appeared in the Zipfian range of only one text do not show up in the Zipfian range of the mixture [20].

Importantly, the overall frequency of the Zipfian domain for very different Chinese texts (mixtures, long texts) is approximately the same and amounts to $\simeq 0.4$; see Tables I and II. In contrast, for English texts this overall frequency grows with the number of different words in the text; see Table III. This is consistent with the fact that for English texts the Zipfian range increases upon mixing.

2. *Emergence of the exponential-like range*

The majority of characters that appear in the Zipfian range of separate texts, but do not appear in the Zipfian range of the mixture, moves to the hapax legomena range of the mixture. Then, for larger mixtures and longer texts, a new, exponential-like range of the rank-frequency relation emerges from within the hapax legomena range.

To illustrate the emergence of the exponential-like range let us start with Fig. 4. Here there are only two short texts mixed and hence the exponential-like range cannot be reliably distinguished from the hapax legomena [21]: for all frequencies with the ranks $r > r_{\max}$ (i.e. for all frequencies beyond the Zipfian range), the number of different characters having exactly the same frequency is larger than 10. (We conventionally take this number as a borderline of the hapax legomena.) However, the trace of the exponential-like range is seen even within the hapax legomena; see Fig. 3.

For bigger mixtures or longer texts, the exponential-like range clearly differentiates from the hapax legomena. In this context, we define r_b as the borderline rank of the hapax legomena: for $r > r_b$, the number of characters having the frequency f_{r_b} is larger than 10. Then the exponential-like range

$$f_r = ae^{-br} \quad \text{with} \quad a < b, \tag{28}$$

exists for the ranks $r_{\max} < r \lesssim r_b$ (provided that $r_{\max}$ is sufficiently larger than r_b); see Table VII. Put differently, the exponential-like range exists from ranks larger than the upper rank $r_{\max}$ of the Zipfian range till the ranks, where the hapax legomenon starts. Tables I, II, VII and Fig. 5 show that the exponential-like range is not only sizable by itself, but (for sufficiently long texts or sufficiently big mixtures) it is also bigger than the Zipfian range. This, of course, does not mean that the Zipfian range becomes less important, since, as we saw above, it carries out nearly 40 % of the overall frequency; see Tables I and II. The exponential-like range also carries out non-negligible frequency, though it is few times smaller than that of the Zipfian and pre-Zipfian ranges; see Tables I, II and VII.

Finally, we would like to stress that we considered various Chinese texts written with simplified or traditional characters, with Modern Chinese or different versions of Classic Chinese; see Tables I, II and Appendix C. As far

[20] As an example, let us consider in detail the mixing of two Chinese texts SBZ and CQF; see Table II. The Zipfian ranges of CQF and SBZ contain, respectively, 306 and 319 characters. Among them 133 characters are common. The balance of the characters upon mixing is calculated as follows: 306 (from the Zipfian range of CQF) + 319 (from the Zipfian range of SBZ) - 133 (common characters) - 50 (characters from the Zipfian range of CQF that do not appear in the Zipfian range of CQF & SBZ) - 54 (characters from the Zipfian range of SBZ that do not appear in the Zipfian range of CQF+SBZ) +27 (characters that enter to the Zipfian range CQF & SBZ from the pre-Zipfian ranges of CQF or SBZ)= 415 (characters in the Zipfian range of CQF & SBZ).

[21] Recall in this context that in the hapax legomena range many characters have the same frequency, hence no smooth rank-frequency relation is reliable.

TABLE VII: Parameters of the exponential-like range (lower and upper ranks and the overall frequency) for few long Chinese texts; see also Tables I and II. Here n is the number of different characters. Recall that the lowest rank of the exponential-like range is $r_{\max}+1$, where $r_{\max}$ is the upper rank of the Zipfian range. The highest rank of the exponential-like range was denoted as r_b; see Tables I and II.

Texts	n	Rank range	Overall frequency
PFSJ	3820	584–1437	0.12816
SHZ	4376	591–1618	0.14317
SJ	4932	536–1336	0.12887
14 texts	5018	626–1223	0.12291

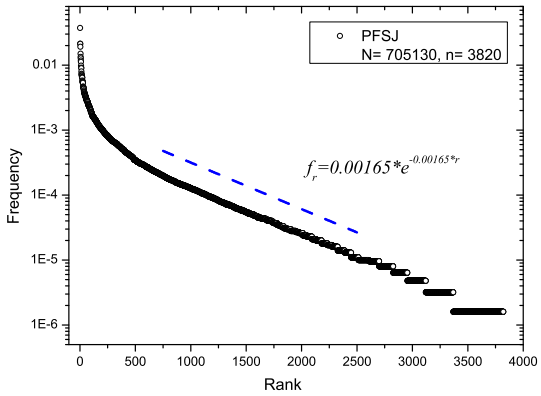


FIG. 5: (Color online) Rank frequency distribution of the long modern Chinese text PFSJ. The exponential behavior $f_r \propto e^{-0.00165r}$ of frequency f_r is visible for $r > 500$. For comparison, the dashed blue line shows a curve $f_r = 0.00165e^{-0.00165r}$. The boundary between the exponential-like range and hapax legomena can be defined as the rank r_b, where the number of words having the same frequency f_{r_b} is equal to 10. For the present example $r_b = 1437$. The Zipf's law holds for ranks $r_{\min} < r < r_{\max}$, where $r_{\max} = 583$, $r_{\min} = 67$; see Table I.

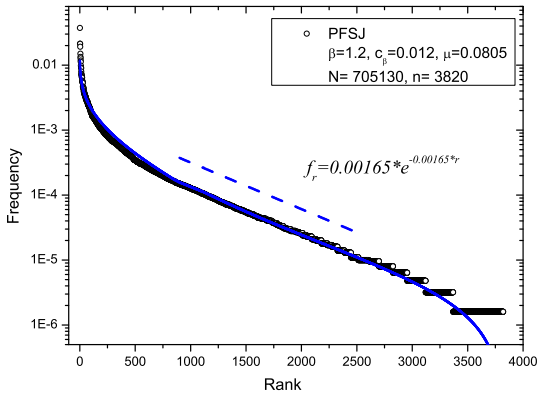


FIG. 6: (Color online) The rank-frequency relation $f(r)$ for characters from the text PFSJ; see Table I. Blue line denotes the numerical solution of (31, 32) at the indicated parameters β and c_β. The dashed blue line indicates at the exponential-like regime.

as the rank-frequency relations are concerned, all these texts demonstrate the same features showing that the peculiarities of these relations are based on certain very basic features of Chinese characters. They do not depend on specific details of texts.

C. Theoretical description of the exponential-like regime

Now we search for a theoretical description for the exponential like regime of the rank-frequency relation of Chinese characters. This description will simultaneously account for the hapax legomena range (rare words) of long Chinese texts.

We proceed with the theory outlined in section III C 1 and III C 2. There we saw that the Zipf's law results from the choice (15) of the prior density for word probabilities $\boldsymbol{\theta}$. Now we need to generalize (15). Recall that the choice of prior densities is is the main problem of the Bayesian statistics [61, 62] [22]. One way to approach this problem is to look for a natural group in the space of events (e.g. the translation group if the event space is the real line) and then define the non-informative prior density as the one which is invariant with respect to the group [61, 62]. Our event space is the simplex $\boldsymbol{\theta} \in \mathcal{S}_n$: the set of n non-negative numbers (word probabilities) that sum to one. The natural group on the simplex is the multiplicative group [61] (in a sense this is the only group that preserves probability relations [62]), and the corresponding non-informative density is the Haldane's prior [61–63] that is given by (14) under

$$u(f) = (n^{-1}c_1 + f)^{-1}, \quad c_1 \to 0. \tag{29}$$

The formal Haldane's prior is recovered from (29) under

[22] We stress that (for a continuous event space) this problem is not solved by the maximum entropy method. In contrast, this method itself does need the prior density as one of its inputs [61].

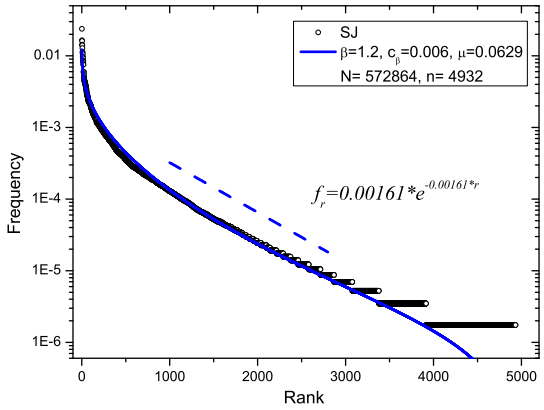


FIG. 7: (Color online) The rank-frequency relation $f(r)$ for characters from the text SJ; see Table II. For parameters and notations see Fig. 6.

$c_1 \equiv 0$; a small but finite constant c_1 is necessary for making the density normalizable.

Note that the prior density (15) which supports the Zipf's law is far from being non-informative. This is natural, because it relates to a definite organization of the mental lexicon [17].

Now the exponential-like regime of the rank-frequency relation can be deduced via a prior density that is intermediate between the Zipfian prior (15) and the non-informative, Haldane's prior (29)

$$u(f) = (n^{-1}c_\beta + f)^{-\beta}, \quad 1 < \beta < 2, \tag{30}$$

where β and $c_\beta > 0$ are to be determined from the data-fitting. Now we can still use (16) but instead of (17, 18) we get the following implicit relations for the smooth part of the rank-frequency relation f_r

$$r/n = \int_{f_r}^{\infty} \mathrm{d}\theta \, \frac{e^{-\mu\theta}}{(c_\beta+\theta)^\beta} \Big/ \int_0^{\infty} \mathrm{d}\theta \, \frac{e^{-\mu\theta}}{(c_\beta+\theta)^\beta} \,, \tag{31}$$

$$\int_0^{\infty} \mathrm{d}\theta \, \frac{\theta\, e^{-\mu\theta}}{(c_\beta+\theta)^\beta} = \int_0^{\infty} \mathrm{d}\theta \, \frac{e^{-\mu\theta}}{(c_\beta+\theta)^\beta}. \tag{32}$$

Figs. 6 and 7 compare these analytical predictions with data. The fit is seen to be good under parameters β and c_β that are not very far from (29). The fact that prior densities close to the non-informative (Haldane's) prior generate an exponential-like shape for the rank-frequency relations is intuitive, since such a shape means that a relatively small group of words carries out the major part of frequency.

As confirmed by Figs. 6 and 7, predictions of (31, 32) that describe the exponential-like regime are not applicable for the Zipfian range.

Importantly, (31, 32) allow to describe the hapax legomena range of long Chinese texts. Following section III D, we equate the solution f_r of (31, 32) to k/N and determine from this $r = \hat{r}_k$: the rank in the hapax legomena range, where the frequency jumps from k/N to $(k+1)/N$. Now $\hat{r}_k$ agrees well with the empiric data for the hapax legomena range of long Chinese texts, and the agreement is better than for the approach based on (24, 25); see Table VIII and cf. with Table VI.

Note that the range of rare words (hapax legomena) relates to that part of the rank-frequency relation which is closest to it, i.e. for long Chinese texts it relates to the exponential-like regime and not to the Zipfian regime.

Though suggestive, the above theoretical results are still preliminary. The full theory of the rank-frequency relations for Chinese characters should really *explain* how specifically a non-Zipfian relations result from mixing texts that separately hold the Zipf's law.

V. DISCUSSION

A. Summary of results

1. As implied by the rank-frequency relation for characters, short Chinese texts demonstrate the same Zipf's law—together with its generalization to high and low frequencies (rare words)—as short English texts; see section III. Assuming that authors write mainly relatively short texts (longer texts are obtained by mixing shorter ones), this similarity implies that Chinese characters play the same role as English words; see Footnote 5 in this context. Recall from section II that *a priori* there are several factors which prevent a direct analogy between words and characters.

2. As compared to English, there are two novelties of the rank-frequency relation of Chinese characters in long texts.

2.1 The overall frequency of the Zipfian range (the range of middle ranks, where the Zipf's law holds) stabilizes at $\simeq 0.4$. This holds for all texts we studied (written in different epochs, genres with different types of characters; see Tables I, II and Appendix C). A similar stabilization effect holds as well for the overall frequency of the pre-Zipfian range for both English and Chinese texts; see Tables I, II and III.

2.2 There is a range with an exponential-like rank-frequency relation. It emerges for relatively longer texts from within the range of rare words (hapax legomena). The range of ranks, where the exponential-like regime holds, is larger than that of the Zipf's law. But its overall frequency is few times smaller; see Tables I, II and VII.

Both these results are absent for English texts; there the overall frequency of the Zipfian range grows with the length of the text, while there is no exponential-like regime: the Zipfian range end with the hapax legomena; see Table III and Fig. 2.

The results **2.1** and **2.2** imply that long Chinese texts do have a hierarchic structure: there is a group of characters that hold the Zipf's law with nearly universal overall frequency equal to $\simeq 0.4$, and yet another group of relatively less frequent characters that display the exponential-like range of the rank-frequency relation.

TABLE VIII: The hapax legomena range for 2 Chinese texts; see Tables I and II; cf. with Table V. We compare the relative errors for, respectively, $\hat{r}_k$ (given by (22)) and $\widetilde{r}_k$ in approximating the data r_k; see section III D 2. Here $\widetilde{r}_k$ is defined by (25, 24). For PFSJ the parameters in (24) are $\gamma = 1.302$ and $b = 0.00013$. For SJ: $\gamma = 1.299$ and $b = 0.00026$. It is seen that the relative error provided by $\hat{r}_k$ is always significantly smaller.

Texts	k	1	2	3	4	5	6	7	8	9	10
PFSJ	$\lvert\widetilde{r}_k - r_k\rvert / r_k$	0.179	0.251	0.294	0.324	0.340	0.356	0.365	0.376	0.387	0.392
	$\lvert\hat{r}_k - r_k\rvert / r_k$	0.020	0.017	0.008	0.001	0.002	0.008	0.008	0.011	0.009	0.011
SJ	$\lvert\widetilde{r}_k - r_k\rvert / r_k$	0.093	0.115	0.136	0.153	0.167	0.176	0.181	0.189	0.195	0.198
	$\lvert\hat{r}_k - r_k\rvert / r_k$	0.020	0.018	0.011	0.001	0.007	0.013	0.0011	0.012	0.008	0.004

B. Interpretation of results

Chinese characters differ from English words, since only long Chinese texts have the above hierarchic structure. The underlying reason of the hierarchic structure is to be sought via the linguistic differences between Chinese characters and English words, as we outlined in section II. In particular, the features **4, 6, 7** discussed in section II can mean that certain homographic content characters play multiple role in different parts of a long Chinese text. They are hence distinguished and appear in the Zipfian range of the long text with (approximately) stable overall frequency $\simeq 0.4$. Since this frequency is sizable, and since the range of ranks carried out by the Zipf's law is relatively small, there is a relatively large range of ranks that has to have a relatively small overall frequency; cf. Tables I, II with Table VII. It is then natural that in this range there emerges an exponential-like regime that is related with a faster (compared to a power law) decay of frequency versus rank.

Recall that the stabilization holds as well for the overall frequency of the pre-Zipfian domain both for English and Chinese texts. The explanation of this effect is similar to that given above (but to some extent is also more transparent): the pre-Zipfian range contains mostly function characters, which are not specific and used in different texts. Hence upon mixing the pre-Zipfian range has a stable overall frequency.

The above explanation for the coexistence of the Zipfian and exponential-like range suggests that there is a relation between the characters that appear in the Zipfian range of long texts and homography. As a preliminary support for this hypothesis, we considered the following construction. Assuming that a mixture is formed from separate texts $T_1, ..., T_k$, we looked at characters that appear in the Zipfian ranges of all the separate texts $T_1, ..., T_k$; see Table II for examples. This guarantees that these characters appear in the Zipfian range of the mixture. Then we estimated (via an explanatory dictionary of Chinese characters) the average number of different meanings for these characters. This average number appeared to be around 8, which is larger than the average number of meanings for an arbitrary Chinese character (i.e. when the averaging is taken over all characters in the dictionary) that is known to be not larger than 2 [47].

We should like to stress however that the above connection between the uncovered hierarchic structure and the number of meanings is preliminary, since we currently lack a reliable scheme of relating the rank-frequency relation of a given text to its semantic features; for a recent review on the (lexical) meaning and its disambiguation within machine learning algorithms see [2].

C. Conclusion

The above discussion makes clear that a theory for studying the rank-frequency relation of a long text, as it emerges from mixing of different short texts, is currently lacking. Such a theory was not urgently needed for English texts, because there the (generalized) Zipf's law (21) describes well both long and short texts. But the example of Chinese characters clearly shows that the changes of the rank-frequency relation under mixing are essential. Hence the theory of the effect is needed.

Finally, one of main open questions is whether the uncovered hierarchical structure is really specific for Chinese characters, or it will show up as well for English texts, but on the level of the rank-frequency relation for morphemes and not the words. Factorizing English words into proper morphemes is not straightforward, but still possible.

Acknowledgments

This work is supported by the Region des Pays de la Loire under the Grant 2010-11967, and by the National Natural Science Foundation of China (Grant Nos. 10975057, 10635020, 10647125 and 10975062), the Programme of Introducing Talents of Discipline to Universities under Grant No. B08033, and the PHC CAI YUAN PEI Programme (LIU JIN OU [2010] No. 6050) under Grant No. 2010008104.

Appendix A: Glossary

• Classic Chinese: (*wén yán*) written language employed in China till the early XX (20th) century. It lost its official status and was changed to Modern Chinese since the May Fourth Movement in 1919. The Modern Chinese keeps many elements of Classic Chinese. As compared to the Modern Chinese, the Classic Chinese has the following peculiarities (1) It is more lapidary: texts contain almost two times smaller amount of characters, since the Classic Chinese is dominated by one-character words. (2) It lacks punctuation signs and affixes. (3) It relies more on the context. (4) It frequently omits grammatical subjects.

• Content word (character): a word that has a meaning which can be explained independently from any sentence in which the word may occur. Content words are said to have lexical meaning, rather than indicating a syntactic (grammatical) function, as a function word does.

• Empty Chinese characters—e.g. "几" *(jǐ)* or "已" *(yǐ)* —serve for establishing numerals for nouns, aspects for verbs *etc*. In contrast, to function characters, they cannot be used alone, i.e. they are fully bound.

• Frequency dictionary: collects words used in some activity (e.g. in exact science, or daily newspapers *etc*) and orders those words according to the frequency of usage. Frequency dictionaries can be viewed as big mixtures of different texts.

• Function word (character): is a word that has little lexical meaning or have ambiguous meaning, but instead serves to express grammatical relationships with other words within a sentence, or specify the attitude or mood of the speaker. Such words are said to have a grammatical meaning mainly, e.g. *the* or *and*.

• Hapax legomena: literally means the set of words (characters) that appeared only once in a text. We employ this term in a broader sense: the set of words (characters) that appear in a text only few times. Operationally, this set is characterized by the fact that sufficiently many words (characters) have the same frequency. Texts written by human subjects contains a sizable hapax legomena. This is a non-trivial fact, since it is not difficult to imagine an artificial text (or purposefully modified natural text) that will not contain at all words that appear only once.

• Homophones: two different words that are pronounced in the same way, but may be written differently (and hence normally have different meaning), e.g. *rain* and *reign*.

• Homographs: two different words (or characters) that are written in the same way, but may be pronounced differently, e.g. *shower* [precipitation] and *shower* [the one who shows]. This example is a proper homograph, since the pronunciation is different. Another example (of both homography and homonymy) is *present* [gift] and *present* [the current moment of time]. Note that the distinction between homographs and polysemes is not sharp and sometimes difficult to make. There are various boundary situations, e.g. the verb *read* [present] and *read* [past] may qualify as homograph, but the meanings expressed are close to each other.

• Homonymes: two words (or characters) that are simultaneously homographs and homophones, e.g. *left* [past of *leave*] and *left* [opposite of *right*]. Some homonymes started out as polysemes, but then developed a substantial difference in meaning, e.g. *close* [near] and *close* [to shut (lips)].

• Heteronyms: two homographs that are not homophones, i.e. they are written in the same way, but are pronounced differently. Normally, heteronyms have at least two sufficiently different meanings, indicated by different pronunciations.

• Key-word (key-character): a content word (character) that characterizes a given text with its specific subject. The operational definition of a key-word (key-character) is that in a given text its frequency is much larger than in a frequency dictionary, which was obtained by mixing together a big mixture of different texts.

• Language family: a set of related languages that are believed (or proved) to originate from a common ancestor language.

• Latent semantic analysis: the analysis of word frequencies and word-word correlations (hence semantic relations) in a text that is based on the idea of hidden (latent) variables that control the usage of words; see [64] for reviews.

• Literal translation: word-to-word translation, with (possibly) changing the word ordering, as necessary for making more understandable the grammar of the translated text. This notion contrasts to the phrasal translation, where the meaning of each given phrase is translated. The literal translation can misconceive idioms and/or shades of meaning, but these aspects are minor for gross (statistical) features of a text, e.g. for rank-frequency relation of its words.

• Logographic writing system is based on the direct coding of morphemes.

• Mental lexicon: the store of words in the long-time memory. The words from the mental lexicon are employed on-line for expressing thoughts via phrases and sentences; see [55] for detailed theories of the mental lexicon. Ref. [55] argues that in addition to mental lexicon humans contain a mental syllabary that is activated during the phonologization of a word that was already extracted from the mental lexicon.

• Morpheme: the smallest part of the speech or writing that has a separate (not necessarily unique) meaning, e.g. *cats* has two morphemes: *cat* and *-s*. The first morpheme can stand alone. The second one expresses the grammatical meaning of plurality, but it is a bound morpheme, since it can appear only together with other morphemes.

• Phoneme: a class of speech sounds that are perceived as equivalent in a given language. An alternative definition: the smallest unit that can change the meaning. Hence normally several different sounds (frequently not distinguished by native speakers) enter in a single

phoneme.

• Pictogram: a graphic symbol that represents an idea or concept through pictorial resemblance to that idea or concept.

• Polysemes: are related meanings of the same word, e.g. the English word *get* means *obtain/have*, but also *understand* (= *have knowledge*). Another example is that many English nouns are simultaneously verbs (e.g. *advocate* [person] and *advocate* [to defend]).

• Syllable: is the minimal phonetic unit characterized by acoustic integrity of its components (sounds), e.g. the word *body* is composed of two syllables: *bo-* and *-dy*, while *consider* consists of three syllables: *con- -si- -der*. In phonetic languages such as Russian the factorization of the word into syllables (syllabification) is straightforward, since the number of syllables directly relates to the number of vowels. In non-phonetic languages such as English, the correct syllabification can be complicated and not readily available to non-experts. Indo-European languages typically have many syllables, e.g. the total number of English syllables is more 10 000. However, 80 % of speech employs only 500-600 frequent syllables [55]. It was argued, based on psycholinguistic studies, that the frequent syllables are also stored in the long-term memory analogously to mental lexicon [55]. The total number of Chinese syllables is much less, around 500 (about 1200 together with tones) [47, 55]. Syllabification in Chinese is generally straightforward too, also because each character corresponds to a syllable.

• Token: particular instance of a word; a word as it appears in some text.

• Type: the general sort of word; a word as it appears in a dictionary.

• Writing system: process or result of recording spoken language using a system of visual marks on a surface. There are two major types of writing systems: logographic (Sumerian cuneiforms, Egyptian hieroglyphs, Chinese characters) and phonographic. The latter includes syllabic writing (e.g. Japanese hiragana) and alphabetic writing (English, Russian, German). The former encodes syllables, while the former encodes phonemes.

Appendix B: Interference experiments

The general scheme of interference experiments in psychology is described as follows [28, 40]. There are two tasks, the main one and the auxiliary one. Each task is defined via specific instructions. The subjects are asked to carry out the main task simultaneously trying to ignore the auxiliary task. The performance times for carrying out the main task in the presence of the auxiliary one are then compared with the performance times of the main task when the auxiliary task is absent. The interference means that the auxiliary task impedes the main one.

There is a rough qualitative regularity noted in many experiments: interference decreases upon increasing the complexity of the main task or upon decreasing the complexity of the auxiliary task.

The most known example of interference experiment is the Stroop effect, where the main task is to call the color of words. The auxiliary task is not to pay attention at the meaning of those words. The experiment is designed such that there is an incongruency between the semantic meaning of the word and its color, e.g. the word *red* is written in black. As compared to the situation when the incongruency is absent, i.e. the word *red* is written in red, the reaction time of performing the main task is sizably larger. This is the essence of the Stroop effect: the semantic meaning interferes with the color perception.

It appears that the Stroop effect is larger for Chinese characters than for English words; see [28] for a review. This is one (but not the only) way to show that getting to the meaning of a Chinese character is faster than to the meaning of an English word.

Another known interference phenomenon is the word inferiority versus word superiority effect. In English these effects amount to the following [65].

If English-speaking subjects are asked to trace out (and count) a specific letter in a text, they make less errors, when the text is meaningless, i.e. it consists of meaningless strings of letters [27, 66]. This is related to the fact that English words are recognized and stored as a whole. Hence the recognition of words—nd moving from one letter to another— interferes with the task of identifying the letter in a single word, and the English-speaking subjects make more errors when tracing out a letter in a meaningful text. This is the word-inferiority effect.

In contrast, if English subjects is presented a single word for a short amount of time, and is then asked about letters of this word, their answers are (statistically) more correct if the word is meaningful (i.e. it is a real word, not a meaningless sequence of symbols). This word-superiority effect is understood by noting that a single word is recalled and/or remembered better due to its meaning.

In contrast to this, Chinese-speaking adults display the word superiority effect, when the naive analogy with English would suggest the word inferiority. They do less errors in tracing out a given character in a string of meaningful characters, as compared to tracing it out in a list of meaningless pseudo-characters [27].

A possible interpretation of this effect is that, on one hand, the definition of Chinese words and their boundaries is somewhat fuzzy, so that the analogue of the English word-inferiority effect is not effective. On the other hand, the Chinese sentence is perceived as a whole, inviting analogies with the English word-superiority effect.

Note that when the Chinese subjects are asked to trace out a specific stroke within a character we expectedly (and in full analogy with the English situation) get that it is easier for Chinese subjects to trace out the stroke in a meaningless pseudo-character than in a meaningful character [27].

Appendix C: A list of the studied texts

1) Two short modern Chinese texts:

- 昆仑殇, *Kūn Lún Shāng* (KLS) by Shu Ming Bi, 1987, (the total number of characters $N = 20226$, the number of different characters $n = 2047$). The text is about the arduous military training in the troops of Kun Lun mountain.

- 阿Q正传, *Ah Q Zhèng Zhuàn* (AQZ) by Xun Lu, 1922, ($N = 18153$, $n = 1553$). The story traces the "adventures" of a hypocrit and conformist called Ah Q, who is famous for what he presents as "spiritual victories".

2) Two long modern Chinese texts:

- 平凡的世界, *Píng Fán de Shì Jiè* (PFSJ) by Yao Lu, 1986, ($N = 705130$, $n = 3820$). The novel depicts many ordinary people's stories which include labor and love, setbacks and pursue, pain and joy, daily life and huge social conflict.

- 水浒传, *Shuǐ Hǔ Zhuàn* (SHZ) by Nai An Shi, 14th century, ($N = 704936$, $n = 4376$). The story tells how a group of 108 outlaws gathered at Mount Liang formed a sizable army before they were eventually granted amnesty by the government and sent on campaigns to resist foreign invaders and suppress rebel forces.

3) Four short classic Chinese texts:

- 春秋繁露, *Chūn Qiū Fán Lù* (CQF), by Zhong Shu Dong, 179-104 BC, (Vol.**1**-Vol.**8**, $N = 30017$, $n = 1661$). A commentary on the Confucian thought and teachings.

- 僧宝传, *Sēng Bǎo Zhuàn* (SBZ), by Hong Hui, 1124, (Vol.**1**-Vol.**7**, $N = 24634$, $n = 1959$). A commentary on the Taoist thought and teachings. Biographies of great Taoist masters.

- 武经总要, *Wǔ Jīng Zǒng Yào* (WJZ), by Gong Liang Zeng and Du Ding, 1040-1044, (Vol.**1**-Vol.**4**, $N = 26330$, $n = 1708$). A Chinese military compendium. The text covers a wide range of subjects, from naval warships to different types of catapults.

- 虎玲经, *Hǔ Líng Jīng* (HLJ), by Dong Xu, 1004, (Vol.**1**-Vol.**7**, $N = 26559$, $n = 1837$). Reviews various military strategies and relates them to factors of geography and climate.

4) A long classic Chinese text:

- 史记, *Shǐ Jì* (SJ), by Qian Sima, 109 to 91 BC, ($N = 572864$, $n = 4932$). Reviews imperial biographies, tables, treatises, biographies of feudal houses and eminent persons.

Appendix D: Key-characters of the modern Chinese text KLS

Here is the list of the key-characters in the Pre-Zipfian and Zipfian range (Table IX) of the modern Chinese text, 昆仑殇 *Kūn Lún Shāng* (KLS) written by Shu-Ming BI in 1987. The text is about the arduous military training in the troops of Kun Lun mountain.

TABLE IX: Key-characters of the modern Chinese text 昆仑殇 *kūn lún shāng* (KLS).

No.	Rank	Character	Pinyin	English	Frequency
1	14	号	*hào*	horn	157
2	32	军	*jūn*	army	86
3	44	兵	*bīn*	soldier	67
4	113	队	*duì*	troop	38
5	118	令	*lìng*	command	37
6	123	部	*bù*	troop	36
7	152	战	*zhàn*	fight/war	28
8	156	命	*mìng*	command	28
9	180	防	*fáng*	protect	24
10	213	血	*xuè*	blood	20
11	216	立	*lì*	stand straight	20
12	224	功	*gōng*	honor	19
13	225	枪	*qiāng*	gun	19
14	252	官	*guān*	officer	16
15	295	锅	*guō*	pan	14
16	299	保	*bǎo*	protect	14
17	300	卫	*wèi*	protect	13
18	352	营	*yíng*	camp	11
19	355	谋	*móu*	strategy	11
20	360	烧	*shāo*	burn	11
21	394	烈	*liè*	martyr	10
22	407	团	*tuán*	regiment	10

Appendix E: Kolmogorov-Smirnov test

The Kolmogorov-Smirnov test (KS test) [67, 68] is used to determine if a data sample agrees with a reference probability distribution. The basic idea of the KS test is as follows.

We need to determine whether a given set X_1, X_2, ... , X_n is generated by i.i.d sampling a random variable with cumulative probability distribution $F(x)$ (null hypothesis). To this end we calculate the the empiric cumulative distribution function (CDF) $F_n(x)$ for X_1, X_2, ... , X_n:

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I_{X_i \leq x}, \tag{33}$$

where $I_{X_i \leq x}$ equals to 1 if $X_i \leq x$ and 0 otherwise. Next we define:

$$D_n = \sup_x |F_n(x) - F(x)|. \tag{34}$$

The advantage of using D_n (against other measures of distance between $F_n(x)$ and $F(x)$) is that if the null hypothesis is true, the probability distribution of D_n does not depend on $F(x)$. In that case it was shown that for $n \to \infty$, the cumulative probability distribution of $\sqrt{n}D_n$

is [67, 68]:

$$P(\sqrt{n}D_n \leq x) \equiv f(x) = 1 - 2\sum_{k=1}^{\infty}(-1)^{k-1}e^{-2k^2x^2}. \quad (35)$$

For not rejecting the null hypothesis we need that the observed value of $\sqrt{n}D_n^*$ is sufficiently small. To quantify that smallness we take a parameter (significance level) α ($0 < \alpha < 1$) and define κ_α as the unique solution of

$$f(\kappa_\alpha) = 1 - \alpha. \quad (36)$$

Now the null hypothesis is not rejected provided that

$$\sqrt{n}D_n^* < \kappa_\alpha, \quad (37)$$

where $\sqrt{n}D_n^*$ is the observed (calculated) value of D_n. Condition (37) ensures that if the null hypothesis is true, the probability to reject it is bounded from below by α. Hence in practice one takes, e.g. $\alpha = 0.05$ or $\alpha = 0.01$.

Note however that condition (37) will always hold provided that α is taken sufficiently small. Hence to quantify the goodness of the null hypothesis one should calculate the p-value p: the maximal value of α, where (37) still holds. For the hypothesis to be reliable one needs that p is not very small. As an empiric criterion of reliability people frequently take $p > 0.1$.

We applied the KS test to our data on the character (word) frequencies; see section III A. The empiric results on word frequencies f_r in the Zipfian range $[r_{\min}, r_{\max}]$ are fit to the power law, and then also to the theoretical prediction described in section III C. With null hypothesis that empiric data follows the numerical fittings and/or theoretical results, we calculated the maximum differences (test statistics) D and the corresponding p-values in the KS tests. From Table X one sees that all the test statistics D are quite small, while the p-values are *much larger* than 0.1. We conclude that from the viewpoint of the KS test the numerical fittings and theoretical results can be used to characterize the empiric data in the Zipfian range reasonably well.

[1] R.E. Wyllys, Library Trends, **30**, 53 (1981).

[2] C.D. Manning and H. Schütze, *Foundations of statistical natural language processing* (MIT Press, 1999).

[3] H. Baayen, *Word frequency distribution* (Kluwer Academic Publishers, 2001).

[4] W.T. Li, Glottometrics, **5**, 14 (2002).

[5] N. Hatzigeorgiu, G. Mikros, and G. Carayannis, Journal of Quantitative Linguistics, **8**, 175 (2001).

[6] B.D. Jayaram and M.N. Vidya, Journal of Quantitative Linguistics, **15**, 293 (2008).

[7] L. Lü, Z.K. Zhang and T. Zhou, PLoS ONE, **5**(12), e14139 (2010).

[8] J. Baixeries, B. Elvevag and R. Ferrer-i-Cancho, PLoS ONE, **8**(3), e53227 (2013).

[9] http://en.wikipedia.org/wiki/Zipf's_law
http://ccl.pku.edu.cn/doubtfire/NLP/Statistical_Approach/Zip_law/references%20on%20zipf%27s%20law.htm

[10] J.B. Estoup, *Gammes sténographique* (Institut Sténographique de France, Paris, 1916).

[11] R. Ferrer-i-Cancho and R. Solé, PNAS, **100**, 788 (2003). M. Prokopenko *et al.*, JSTAT, P11025 (2010).

[12] B. Mandelbrot, *An information theory of the statistical structure of language*, in *Communication theory*, ed. by W. Jackson (London, Butterworths, 1953).
B. Mandelbrot, *Fractal geometry of nature* (W. H. Freeman, New York, 1983).

[13] B. Corominas-Murtra *et al.*, Phys. Rev. E, **83**, 036115 (2011).

[14] D. Manin, Cognitive Science, **32**, 1075 (2008).

[15] G.A. Miller, Am. J. Psyc. **70**, 311 (1957). W.T. Li, IEEE Inform. Theory, **38**, 1842 (1992).

[16] M.V. Arapov and Yu.A. Shrejder, in *Semiotics and Informatics*, v. 10, p. 74 (Moscow, VINITI, 1978). I. Kanter and D. A. Kessler, Phys. Rev. Lett. **74**, 4559 (1995). B.M. Hill, J. Am. Stat. Ass. **69**, 1017 (1974). G. Troll and P. beim Graben, Phys. Rev. E **57**, 1347 (1998). A. Czirok *et al.*, *ibid.* **53**, 6371 (1996). K. E. Kechedzhi *et al.*, *ibid.* **72** (2005).

[17] A.E. Allahverdyan, Weibing Deng and Q.A. Wang, Phys. Rev. E **88**, 062804 (2013).

[18] D. Howes, Am. J. Psyc. **81**, 269 (1968).

[19] R. Ferrer-i-Cancho and B. Elveva, PLoS ONE, **5**, 9411 (2010).

[20] K.H. Zhao, Am. J. Phys. **58**, 449 (1990).

[21] R. Rousseau and Q. Zhang, Scientometrics, **24**, 201 (1992).

[22] D.H. Wang *et al.*, Physica A, **358**, 545 (2005).

[23] S. Shtrikman, Journal of Information Science, **20**, 142 (1994).

[24] Le Quan Ha *et al.*, *Extension of Zipf's Law to Words and Phrases*, Proceedings of the 19th international conference on Computational linguistics, **1**, pp. 1-6, (2002).

[25] Q. Chen, J. Guo and Y. Liu, Journal of Quantitative Linguistics, **19**, 232 (2012).

[26] D. Aaronson and S. Ferres, J. Memory and Language, **25**, 136 (1986).

[27] H.C. Chen, *Reading comprehension in Chinese*, in H.C. Chen & O. J. L. Tzeng (Eds.), Language processing in Chinese (pp. 175- 205). Amsterdam, Elsevier, 1992.

[28] R. Hoosain, *Speed of getting at the phonology and meaning of Chinese words*, in *Cognitive neuroscience studies of Chinese language*, H.S.R. Kao, C.K. Leong and D.G. Gao (eds.) (Hong kong University Press, Hong kong, 2002).

[29] G.K. Zipf, *Selected studies of the principle of relative frequency in language.* (Harvard University Press, Cambridge MA, 1932).

[30] L. Lü, Z.K. Zhang and T. Zhou, Sci. Rep. **3**, 1082 (2013).

[31] C.K. Hu and W.C. Kuo, *Universality and Scaling in the Statistical Data of Literary Works*, POLA Forever, 115-139 (2005).

[32] J. Elliott *et al.*, *Language identification in unknown signals*, Proceedings of the 18th conference on Computa-

TABLE X: Kolmogorov-Smirnov test (KS test) for the fitting quality of our results (texts are defined in Tables I and II). In the KS test, D and p denote the maximum difference (test statistics) and p-value respectively. D_1 and p_1 are calculated from the KS test between empiric data and numerical fitting, D_2 and p_2 are between empiric data and theoretical result, D_3 and p_3 are between numerical fitting and theoretical result; see section III A. Note that for making the testing even more vigorous the presented results for the KS characteristics are obtained in the original coordinates (2); similar results are obtained in logarithmical coordinates (3) that are employed for the linear fitting.

Texts	D_1	p_1	D_2	p_2	D_3	p_3
TF	0.0418	0.865	0.0365	0.939	0.0381	0.912
TM	0.0529	0.682	0.0562	0.593	0.0581	0.568
AR	0.0564	0.624	0.0469	0.783	0.0443	0.825
DL	0.0451	0.812	0.0421	0.865	0.0472	0.761
AQZ	0.0586	0.587	0.0565	0.623	0.0601	0.564
KLS	0.0592	0.578	0.0641	0.496	0.0626	0.521
CQF	0.0341	0.962	0.0415	0.863	0.0421	0.857
SBZ	0.0461	0.796	0.0558	0.635	0.0616	0.538
WJZ	0.0427	0.852	0.0475	0.753	0.0524	0.691
HLJ	0.0375	0.923	0.0412	0.875	0.0425	0.862

tional linguistics, **2**, pp. 1021-1025 (2000).
J. Elliot and E. Atwell, Journal of the British Interplanetary Society **53**, 13 (2000).

[33] H.P. Luhn, IBM J. Res. Devel. **2**, 159 (1958).

[34] S.M. Huang *et al.*, Decision Support Systems, **46**, 70 (2008).

[35] D.M.W. Powers, *Applications and explanations of Zipf's law*, in D.M.W. Powers (ed.), New Methods in Language Processing and Computational Natural Language Learning (NEMLAP3/CONLL98), ACL, 1998, pp. 151-160.

[36] G. Sampson, Linguistics, **32**, 117 (1994).

[37] J. DeFrancis, *Visible Speech: the Diverse Oneness of Writing Systems* (University of Hawaii Press, Honulu, 1989).

[38] J. L. Packard, *The Morphology of Chinese: A linguistic and cognitive approach* (Cambridge University Press, Cambridge, 2000).

[39] K. Turner, *Visualizing Zipf's Law in Japanese*, available at this link:
http://classes.soe.ucsc.edu/cmps161/Winter12/projects/katurner/proj/paper/paper.pdf

[40] R. Hoosain, *Psychological reality of the word in Chinese*, in H.C. Chen and J.L. Tseng (eds.), Language processing in Chinese, pp. 111-130, (Amsterdam, Netherlands, 1992).

[41] I.M. Liu *et al.* Chinese Journal of Psychology, **16**, 25 (1974).

[42] S.H. Hsu and K.C. Huang, Perceptual and Motor Skills, **91**, 355 (2000); *ibid.* **90**, 81 (2000).

[43] X. Luo, *A maximum entropy Chinese character-based parser*. Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, 2003.

[44] Wm. C. Hannas, *Asia's Orthographic Dilemma* (University of Hawaii Press, 1997).

[45] C.Y. Chen *et al.*, *Some distributional properties of Madanrin Chinese*, Proceedings of the first Pasific Asia conference on formal and computational linguistics, p. 81 (Taipei, 1993).

[46] http://myweb.tiscali.co.uk/wordscape/wordlist/homogrph.html

[47] N.V. Obukhova, Quantitative linguistics and automatic text analysis (Proc. of Tartu university), **745**, 119 (1986).

[48] N.J.D. Nagelkerke, *A Note on a General Definition of the Coefficient of Determination*, Biometrika, **78** (3), 691 (1991).

[49] M. L. Goldstein, S. A. Morris, G. G. Yen, *Eur. Phys. J. B*, **41**, 255 (2004).

[50] H. Bauke, *Eur. Phys. J. B*, **58**, 167 (2007).

[51] A. Clauset, C. R. Shalizi and M. E. J. Newman, *SIAM Rev.*, **51**, 4 (2009).

[52] R.E. Madsen *et al.*, *Modeling word burstiness using the Dirichlet distribution*, in Proc. Intl. Conf. Machine Learning, 2005.

[53] S. Bernhardsson, L. E. Correa da Rocha, P. Minnhagen, Physica A **389**, 330 (2010); New J. Phys. **11**, 123015 (2009).

[54] T. Hofmann, *Probabilistic Latent Semantic Analysis*, in Uncertainty in Artificial Intelligence, 1999.

[55] W.J.M. Levelt *et al.*, Beh. Brain Sciences, **22**, 1 (1999).

[56] J. Tuldava, Journal of Quantitative Linguistics **3**, 38 (1996).

[57] D. Krallmann, *Statistische Methoden in der stilistischen Textanalyse* (Inaug.-Dissert. Bonn, 1966).

[58] S.K. Baek, S. Bernhardsson and P. Minnhagen, New Journal of Physics **13**, 043004 (2011).

[59] Y. Dover, Physica A **334**, 591 (2004).

[60] E.V. Vakarin and J. P. Badiali, Phys. Rev. E **74**, 036120 (2006).

[61] E.T. Jaynes, IEEE Trans. Syst. Science & Cyb. **4**, 227 (1968).

[62] M. Jaeger, Int. J. Approx. Reas. **38**, 217 (2005).

[63] J. Haldane, Proceedings of the Cambridge Philosophical Society, **28**, 55 (1932).

[64] T. Hofmann, *Probabilistic Latent Semantic Analysis*, in Uncertainty in Artificial Intelligence, 1999.

[65] A.F. Healy and A. Drewnowski, Journal of Experimental Psychology: Human Perception and Performance **9**, 413 (1983).

[66] *Reading Chinese Script: A Cognitive Analysis*, edited by J. Wang, A.W. Imhoff and H.-C. Chen (Lawrence Erl-

baum Associates, New Jersey, 1999).
[67] A.N. Kolmogorov, Giornale dell' Instituto Italiano degli Attuari, **4**, 77 (1933).
[68] P.T. Nicholls, *J. Am. Soc. Information Sci.*, **40**, 379 (1989).

Statistical Structure Modeling and Optimal Combined Strategy based Chinese Components Recognition

Bowen Yu, Xiaohui Liang, Jiajia Hu, Linjia Sun
State Key Laboratory of Virtual Reality Technology and Systems
Beihang University
Beijing, China
Email: {yubw, lxh, hjj, sunlinjia}@vrlab.buaa.edu.cn

***Abstract*—Extracting perceptually meaningful components plays an essential role in Chinese character studying process. This paper proposes an improved statistical structure modeling method to pick up all meaningful components in one character. Each stroke is represented by the distribution of the feature points both in model component and input character. The stroke relations are effectively reflected by the statistical dependency. The mutual information among strokes can be calculated to measure the importance of relationships. Considering the local features of components' difference from the whole character recognition, this paper proposes a method based on local feature to select local components rather than the whole character. At last, we adopt optimal combined strategy to select the best component recognition result. By this method, all the components in one character can be achieved.**

***Keywords*-Chinese component recognition; statistical structure modeling; neighbor selection; optimal combined strategy**

I. Introduction

Chinese writing, as the representative of ideographic systems, is not based on the irreducible elements used in speaking (syllabic or alphabetic) like phonetic system. In Chinese information processing, each character is treated as a separate symbol, which brings a major problem of mass encoding set. According to Structure Study of Chinese Ideography, the elements of Chinese writing are called components which brought in certain meanings when they were used to construct characters. For example, those characters like "桃", "李", which have a common component "木", are all used to record a name of "tree". Therefore, extracting perceptually meaningful components plays an essential role in the study of Chinese writing as well as the information processing of Chinese characters.

The Chinese recognition problem includes on-line recognition and off-line recognition. In on-line recognition, the trajectories of pen-tip movements and pen-up/down switching information are recorded for recognition. On the other hand, for off-line character recognition, the only available information is the input image. Comparing to the wide use of on-line recognition, off-line recognition is still a difficult problem.We have to extract necessary features from the input character image for recognition. In this paper, we only refer to the knowledge of off-line character recognition.

Generally, the recognizer represents and analyzes the Chinese character image by statistical or structural methods. The statistical methods usually process the image to form a feature vector so that their similarity can be calculated by the distance between two vectors. On the other hand, the structural methods extract the structure information from the image which the relationship between strokes are emphasized. The structural recognizer regards the relationship between strokes as the most stable feature of one character.

Various statistical methodologies have been proposed for character recognition, such as k-nearest-neighborhood classifer[1], K-Means clustering and Gaussian distribution selector[2], nonlinear active shape models[3], contextual vector quantization[4] and Mahanalobis distance[5].

In structural character recognition method, character are often decomposed into parts to analyze. For instance, the part of characters can be contours, strokes or hierarchical models. The structural matching method is approximately equal to stroke matching.

As the basic component of Chinese character, the description of stroke is very important in structural matching. However, the natural strokes are difficult to extract in off-line character recognition. Instead of the natural strokes, straight line segments are more frequently used as the structural primitive. Also the line between feature points including end points and junction points is another form of stroke[6] .

In order to extract more information in stroke, the strokes are categorized into several types, such as horizontal, vertical, slash, back-slash, dot and hook[7]. In these methodologies, all the strokes are classified into the predefined stroke types where each type has different tolerances of orientation, angles and length. However, this method depends heavily on the heuristic knowledge because the process of manually design.

Another essential element in stroke modeling is relationship between strokes. Many researchers regard the relationship between strokes as the most important structural information because stroke relationship is more stable than stroke itself. The relation information can also distinguish two characters which are very similar in shape.

There are both advantages and disadvantages in two meth-

978-0-7695-4911-8/12 $26.00 © 2012 IEEE
DOI 10.1109/SITIS.2012.43

ods. The statistical methods such as k-nearest-neighborhood classifier, K-Means clustering and Gaussian distribution selector and contextual vector quantization, they all perform better when process the image with noises. However, they reflect the character structure indirectly, thus have difficulties to distinguish characters with similar shapes such as "王" and "玉". However, the structure methods are more adaptable on the similar characters. In this method, characters are decomposed into parts to analyze. The component of characters can be contour, strokes, and hierarchical models. Most of structural matching methods adopt strokes as component, structural matching is approximately equal to stroke matching so far.

For the special application such as component recognition and segmentation, they both can't solve the recognition problem perfectly[8] . The statistical methods tend to focus on the global features, while the component recognition always needs local features. And the structure methods can't deal with nonstandard characters because of the existence of unexpected stroke segment.

However, after the appearance of the methods combined with statistical and structural methods is proposed, the defects mentioned above could be fixed. It mainly uses various kinds of statistical methodologies to describe the structure of Chinese characters in the images. In the special application of component recognition, the combined methods could achieve better performance. Compared with pure statistical method, the structure of Chinese character is emphasized to achieve better recognition rate. In comparison with pure structure method , the combined method's statistical modeling of strokes have more strong resistance to noise which allows us to process not only the regular character images[9].

In combined method, the stroke is represented by the distributions of its length, position and angle[10]. The model stroke is represented by the joint distribution of its feature points, while the character is represented by the joint distribution of the component strokes. The position and the shape of strokes are statistically modeled and their distributions are estimated from training samples.This statistical modeling is adopted to represent both statistical and structural information of characters. Subsequently the MRF-based methods are proposed by Jia Zeng[11], which makes use of the energy of cliques to measure the similarity between the target character and the model character. It provides a much more enriched vocabulary to describe character structures. The difference between different characters can be emphasized by the special design of neighborhood system and clique potentials.

We mainly focus on transferring the combined methods from Chinese character recognition to Chinese component recognition. By combined method, the strokes in input character could be modeled to reflect both statistical and structural information, then the recombination of input strokes could realize the target of component recognition.

There are three main contributions in our work:

1) We improved the combined method, which separates the feature extraction into two parts. This improved method could emphasize the local feature for component recognition and be regardless of the same component in different position of different characters.

2) Sampling the feature points of the stroke to reduce the influence caused by different scale of input character.

3) Adopting the optimal combined strategy to select the best component recognition result.

II. Chinese Component Modeling

A. Statistical Structure Modeling

In statistical structure modeling method, the component and the whole character are both represented by strokes. The stroke of the model components is manually labeled for the component recognition in character images. The labeling process is to manually combine the sub-strokes extracted by the initial extraction algorithm. The strokes are assumed to obey the multivariate normal joint distribution $X \sim N(\mu, \Sigma)$, and their attributes could be represented by attributes of points belong to them, where we use Gabor filter to measure direction information of the these points[12]. Given the input character image S, the formula 1 is utilized to measure the similarity between S and the model component C. The similarity can be measured by the joint probability of all matching instances between s_i and r_i, where s_i represents the strokes in input character and r_i represents the strokes in model component.

$$Pr(S = C) \equiv Pr(s_1 = r_1, s_2 = r_2, ..., s_n = r_n). \tag{1}$$

However, recognizing the character with joint distribution of all component strokes is too complicated. To solve this problem, the conditional probability is applied.

A multivariate normal joint distribution $X \sim N(\mu, \Sigma)$ is composed of sub-distributions $X_A \sim N(\mu_A, \Sigma_A)$ and $X_B \sim N(\mu_B, \Sigma_B)$ as follows:

$$X = \begin{bmatrix} X_A \\ X_B \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix}. \tag{2}$$

The parameters of conditional distribution $Pr(X_B|X_A = x_A)$ are computed by the following formula:

$$\mu_{B|A} = \mu_B + \Sigma_{BA}\Sigma_{AA}^{-1}(x_A - \mu_A). \tag{3}$$

$$\Sigma_{B|A} = \Sigma_{BB} - \Sigma_{BA}\Sigma_{AA}^{-1}\Sigma_{AB}. \tag{4}$$

Then the matching probability $Pr(s_1 = r_1, s_2 = r_2)$ can be calculated by conditional probability.

B. Neighbor Selection and Local Feature Extraction

The joint distribution contains all information about the strokes and their relationships without the consideration of their importance. In fact, reflecting all relationships is not necessary both in time complexity and the proper accuracy.

Therefore, we tend to select the necessary relationship to simplify the relationship model.

The joint probability can be transformed to every conditional probability multiplied together, which affords us to calculate the joint probability by the conditional probability. The equation is as followed:

$$\begin{aligned} Pr(S) &= Pr(s_1, s_2, ..., s_n) = Pr(s_1)Pr(s_2, ..., s_n|s_1) \\ &= ... = \prod_{i=1}^{n} Pr(s_i|s_1, s_2, ..., s_{i-1}). \end{aligned} \tag{5}$$

Using the equation above, we can calculate the joint probability by conditional probability of each stroke. However, we have to calculate the dependencies from all preceding strokes in order to get one conditional probability. The neighborhood system can simplify this problem by selecting some important relationship to calculate the conditional probability. This approximation is acceptable both on accuracy and time cost. The approximated equation is shown below:

$$Pr(S) \approx \prod_{i=1}^{n} Pr(s_i|nei(s_i)). \tag{6}$$

The joint distribution contains all information of the strokes and their relationships. Owing to the complexity of reflecting all mutual relationships, we tend to select the most important relationship by minimizing the Kullback-Leibler measure to simplify the relationship model. The details of derivation process can be found in reference [10].

Only one stroke is selected as the neighbor of each stroke. According to the specialty of the data, the $M(s_i; s_j)$ tends to be positive if two strokes are tending to be parallel. To the contrary, the $M(s_i; s_j)$ tends to be negative if two strokes are tending to be vertical. Also, the longer one stroke is, the more stable feature it has, and we finally choose the strokes which have maximum absolute value as the neighbor (see Fig. 1).

Here, we put more concentration on the extraction of the local feature. We decided to calculate the bounding box of two strokes which are neighbors to each other. Then the position relation, the relative length ratio are all extracted, as shown in Fig. 2. These features could effectively reflect the local characters of component, which allows us to better extract the Chinese component in follow-up work.

At last, the individual and neighbor information every stroke has in one Chinese character component are stored in database, where the database could be normal "txt" or binary file and so on.

III. Chinese Components Recognition System

A. Candidate Stroke Extraction

First, we use skeletonizing algorithm to thin the images. Second, the cross points and the end points are both extracted for searching the basic strokes. When the component images in database are processed, the manual interaction can be adopted to combine some strokes in order to reduce

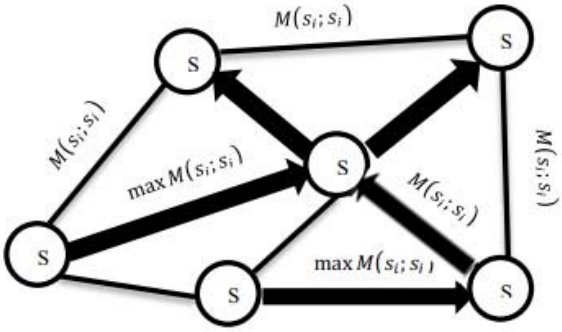


Figure 1. Finding optimal set of the neighbor relationships.

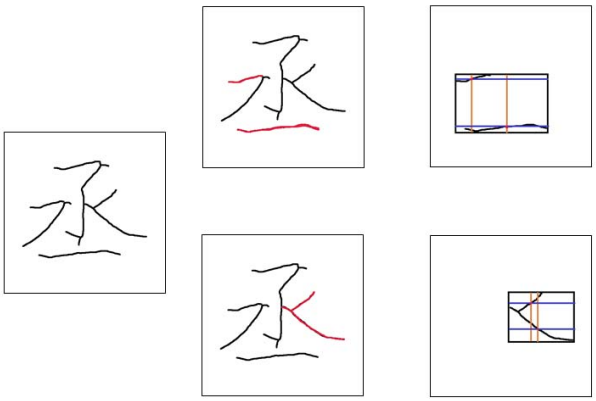


Figure 2. Local features extraction between neighbor strokes.

the number of model strokes. On the process of character image which is input by users to recognize the components, the basic strokes should be combined into candidate strokes when we try to find the proper component. The algorithm is shown in algorithm 1. The array called record is used to write down the details about which strokes the current combination of stroke is consisted of. After the algorithm 1 is carried out, all candidate strokes to the proper component are produced. For example, when we input the Chinese character "ཊ", comparing with the component "ཀ", the candidate stroke extraction results are shown in Fig. 3.

Because of the difference both in shape and scale between the single component and the component in Chinese character, using the traditional calculating method can not achieve preferable accuracy. In order to deal with this problem, we should sample the stroke data. After the Gabor filter is adopted, we firstly decide to select key points in the stroke like end points, corner points and cross points. Then we further sample the points between the key points. The equal number of points are sampled between key points compared to the component in database. Referring to the stroke of the

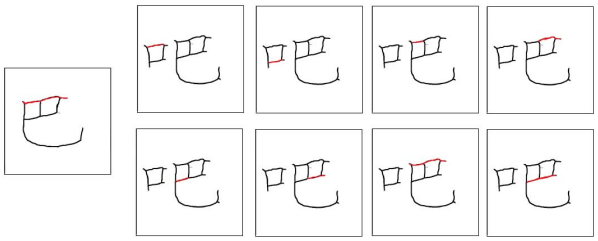


Figure 3. Candidate stroke extraction.

Input: All basic strokes detected by the stroke extraction algorithm.
Output: The candidate strokes in character image to one model stroke in Chinese component.

Initialization:
  We keep input strokes which are either similar in direction or rather small in open list.
  Build up one vector to record the information of combination.
Extension:
for $i = 0$ to the size of similar strokes **do**
  record[i] = MAX; j = i+1;
  while record[i]!=-1 **do**
    if stroke i,j are connected and $j < size$ **then**
      Combine them and push the new stroke into candidate strokes;
      Update the stroke i, then record j in record[i+1];
      i = i + 1;
    else if $j < size$ **then**
      j = j + 1;
      Continue;
    end if
    i = i - 1;
    j = record[i] + 1;
    record[the number of combination/2 + i-1]=-1;
  end while
end for

Algorithm 1: Candidate Stroke Extraction

model component, we could calculate the similarity between the sample data and the model stroke. By sampling the data, we reduce the differences caused by different scales as much as possible. In our experiment, we find that this method could produce better results than original one.

B. Matching Algorithm

The whole stage is divided into two sub-stages, single component matching and all components matching. The first problem can be solved by heuristic search algorithm. The aim of single component matching is to find out the best set of strokes in input character to every model component in database. The second problem is categorized as the variant knapsack that demands us to find the best combination of the component found by the first stage, which could fulfill the knapsack as many as possible without stroke usage collision. This is the optimal combined strategy we mentioned above.

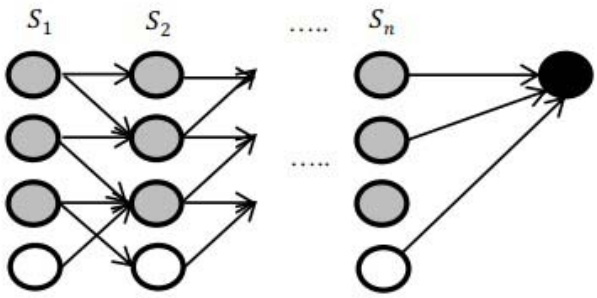


Figure 4. Finding optimal matching.

1) Single Component Matching: The heuristic search algorithm is applied to match the strokes of character to every stroke of the model component. The matching graph should be firstly built up by the model strokes in component. Since the longer strokes represent the more stable structure information, they are sequenced in front. The strokes which are the neighbor of several strokes in component is also put in front.

In this stage, a matching result is defined by a set of correspondences between the model strokes and their matching instances. The matching stage is illustrated in Fig. 4. The S_i represents the model strokes in the model component. The gray nodes represent the candidate stroke which is calculated by the algorithm 1. The white ones show that the optimal candidate stroke s_i for model stroke r_i is not found. The last black node records the final result of this matching process, which the component will be selected if the result is greater than current best record, otherwise, discarded. At last, for one model component, we could find the best set of strokes in input character which indicates the combination of these strokes are most similar to the certain component in database. The algorithm is shown in algorithm 2, which the record[i] here illustrates the selection in candidate strokes generated by algorithm 1 for the certain component.

During the stroke matching process, for example, in stage S_i, if stroke i's neighbor is matched before, then the conditional probability is calculated instead of the common probability. Otherwise, the common probability is used if the neighbor of S_i is not matched. Two input strokes s_i and s_j, whose matching goal are r_i and r_j, respectively, are calculated by conditional probability if they satisfy the following conditions:

(1) s_i is already matched with r_i.
(2) r_i is the neighbor to r_j.

Input: Candidate strokes to all strokes of the model component.
Output: The best set of strokes that match the model component.

```
Procedure:
while the path are not all found do
    if candidate stroke j is selected as the model stroke i and i < size of model stroke then
        i=i+1;
        j=0;
    else if i=0 then
        break;
    else
        i=i-1;
        j=record[i]+1;
    end if
end while
select best path which has the maximal similarity.
```

Algorithm 2: Searching Algorithm

Otherwise, the probability in stage j is calculated without the conditional probability.

Here, we prefer to consider not only the conditional probability but also the local feature extracted in Section 2. The local feature should be emphasized when the graph is searched for the component. To emphasize the local feature, we should utilize the ratio of relative position and length calculated in the training stage. Both the local features and the conditional probability are applied to promote the performance of component recognition.

2) All Components Matching: Here, we modify the matching result in last section, in order to deal with the Chinese character like "磊", "炎", which has reduplicate component. If there is another result which has absolutely different strokes usage that is also similar to the model component in one searching process, for example, the "木" exists in "森", not only one searching result produced in stroke matching is saved.

In fact, the single component matching is prepared for this subsection. After the single component matching stage, we could obtain possible components composed of many kinds of combination of the strokes in input characters to different component. The different combination results we all record have different input strokes usage. We define the best recognition result as a set of possible components, which the possible components are different in input strokes usage, as well, minimize the unselected strokes of strokes in input characters. We regard this problem as variant knapsack problem. The knapsack's size is equal to the size of strokes in input character. Every matching result has the attribute to describe the usage of input strokes. Then the problem can be classified into knapsack problem, which we aim

Input: Every match result of model component.
Output: The components which consist of the character in input image.

```
Procedure:
for j = 1 to size of input strokes do
    if j > c[n] then
        f(1, j) = v[j];
    else
        f(1, j) = 0;
    end if
end for
for i = 1 to n do
    for j = 1 to size of input strokes do
        if j >= c[i] then
            f(i, j) = max
            [f(i − 1, j), f(i − 1, j − c[i]) + v[i]];
        else
            f(i, j) = f(i − 1, j);
        end if
    end for
end for
```

Algorithm 3: Component Recognition Algorithm

to find the best combination in those result produced by the single matching problem to fulfill the knapsack without stroke usage collision as many as possible. The algorithm is shown in algorithm 3, where $f(i, j)$ indicates the best value under the knapsack condition j, when component i to n can be selected. And c,v indicates stroke usage and the value of stroke usage respectively. The ">" and "<" in algorithm 3 means the knapsack could afford matching result i without collision to other existed strokes in knapsack or not. Through the solution of variant knapsack problem, we could finally obtain the best component combination in input character and the aim of component recognition is able to be completed.

IV. Experimental Results

In this section, we will introduce our experiment in five sub-parts. The first subsection will introduce the process of sampling data, in order to reduce the difference caused by different scales. The content in second subsection includes the neighbor selection and conditional probability. The third subsection shows the result of component recognition. The forth and fifth parts will give out the contrast experiment and statistical data respectively.

A. Sample the Data

As mentioned above, we define the stroke as the distribution of feature points. Here, the feature points are extracted from all points in one stroke. In order to reduce the influence caused by the different scales, we decide to sample the points both in training stage and recognizing stage. The end points,

Figure 5. The feature points in sampling process.

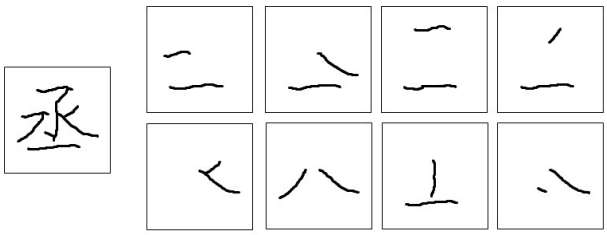


Figure 6. The result of neighbor selection.

crossing points and corner points are extracted firstly, then sample the data between these points according to the ratio of length. The whole sample stage complies with the rule that ensure the equal number of point data. The details are shown in Fig. 5 .

In the example "七" shown in Fig. 5. We firstly extract the end points, crossing points and corner points. At the base of these points, we further sample the points between these special points. The same number of sample points should be extracted in the same character with different size. The sampling process could obviously promote the accuracy rate in single stroke matching.

B. Neighbor system and local feature

The example of neighbor selection are shown in Fig. 6. The picture shown all presents the most possible neighbor by the calculation illustrated in section 3. This process is applied in the training stage, where the component are manually combined and the neighbors are needed to be decided. The relationship is either tending to be parallel or vertical; otherwise the neighbor stroke's length is rather long which indicates it is more stable. Particularly, it successfully detects high-level relationships, which makes one ideal foundation for the recognition in next part.

In Fig. 7, the conditional probability of model stroke is visualized to test whether the statistical dependency could reflect the stroke relationships well. First, we select one stroke with two neighbors by the methods illustrated above. Then the user draws one stroke according to the model, which makes some changes at the location, length or angle,

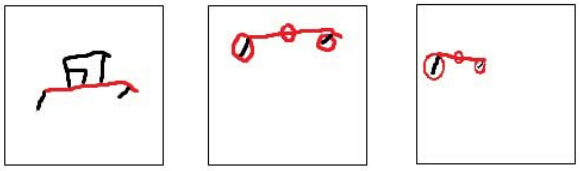


Figure 7. Conditional probabilities given the certain neighbors.

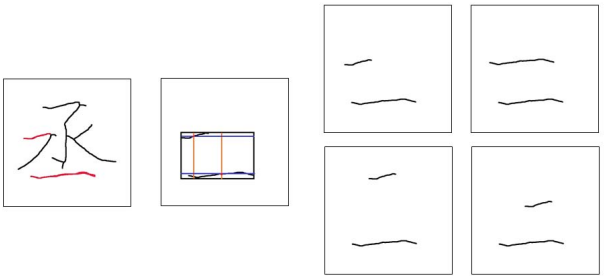


Figure 8. Stroke matching using local features.

to test whether the stroke can be matched to the right model stroke.

The local features are extracted between the strokes and their neighbors. We record the ratio of length and relative position for the component recognition.When the matching stage is processing, we calculate the relative ratio of length and position to measure the similarity .The local features are utilized in Fig. 8. Compared with the first picture in last four pictures, the relative length ratio, the center position and the center distance are not matched so well, respectively. The import of this kind of local feature could emphasize more on the relation between strokes in one component. Finally, we see that the probability we choose the first one as the matching result is much more than others.

C. Component Recognition

The optimal combined strategy based recognition process are shown in Fig. 9. Here, the character "人", "土" and "田" are collided in input strokes usage, so they could not be selected together. As the result, the optimal combined strategy makes the best result of "田" and "木".

The model-driven component recognition are shown in Fig. 10.The first column shows the original character image. The second column shows the result after image thinning process, where the strokes are all extracted for the candidate stroke selection. The third and fourth column show recombination of the candidate stroke driven by the model component to realize the purpose of component recognition and segmentation. Then the target components are extracted by this method. Even the structure information can be further estimated by the distribution of the components which are found by the algorithm.

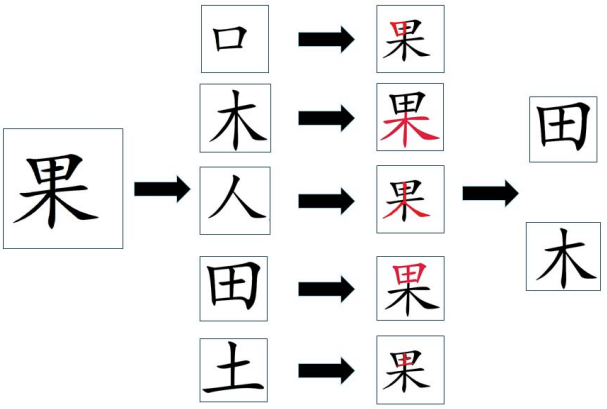


Figure 9. Optimal combined strategy based component recognition.

Figure 10. The results of component recognition.

D. Contrast Experiment

We firstly compare our method with some other efforts we have tried. By comparison, the method based on the connected area appears to have over-segment problem, which always divides the character into fragmentary pieces. The recombination of these pieces is both time-consuming and difficult to find the correct result. The more single areas there are, the harder the recombination is. The Viterbi algorithm segmentation using HMM search for path from one side to the other, could segment the character into different parts. After the initial segmentation, the whole character recognition method could be applied to recognize the component. But the threshold is rather difficult to determine. Different threshold causes different results, as shown in Fig. 11. The comparison results show that this kind of method does heavily depend on the segmentation results, where our method does not. And the segmentation is built up on the

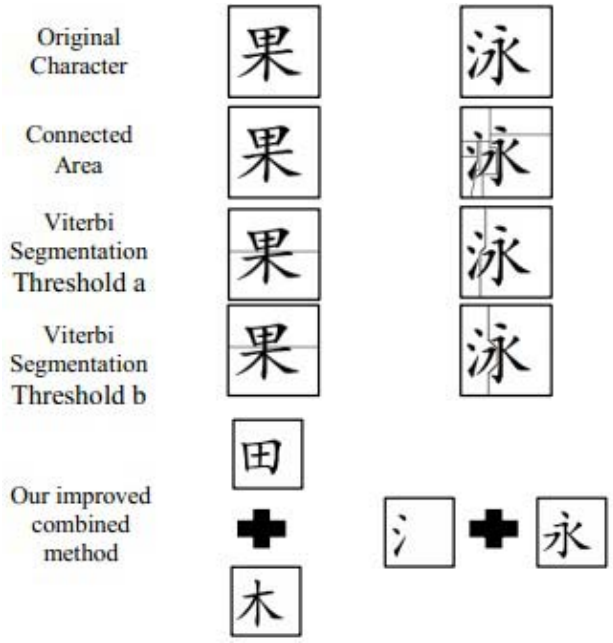


Figure 11. Comparison with the methods which recognize components after segmentation.

prior knowledge about the character's structure (left and right, up and down, etc.), which decides the segmentation is vertical, horizontal or others. On the one hand, the statistical structure method combined with dynamic window is not affordable on the complexity. On the other hand, it could not deal with the special structure called sub-surround structure and surround structure such as "病" and "国".

From the experiment shows above, the method which firstly segment and then recognize can not produce satisfactory result all the time. Our method deals with the segmentation and recognition at the same time, which could produce rather preferable result.

Furthermore, we decide to compare our proposed method with some representative other works on radical recognition. The comparison result are shown in table I.

1) Method 1: Using the statistical structure modeling and optimal combined strategy based method described above, experiments were conducted on 100, 200 radicals covering 500 and 1000 Chinese characters, named experiment 1 and 2 respectively. Here, we only has single copy for each component regardless of their different position in different characters in database.

2) Method 2: Using the nonlinear PCA approach. The experiments were conducted on the same test set as for Method 1. Here we stress this method should train large number of components, so that it could recognize the same component in different position of different characters. For instance, the same component "木" has different position in "杉", "沐", "呆" and "杏". Our method could avoid

Table I
THE CONTRAST EXPERIMENT OF COMPONENT RECOGNITION

	Experiment 1	Experiment 2
Method 1(Our Method)	96.8	92.6
Method 2(Nonlinear ASM)	96.4	89.3
Method 3(Stroke-Based)	92.6	86.4

this by the statistical structure modeling, which we only have one copy for one component without the consideration of its different position in different characters, the whole recognition process allows us to recognize the component in different position of different characters.

3) Method 3: The stroke-based approach of Wang and Fan[13]. Their method's performance depends heavily on the segmentation result in first stage, which made their method hard do with the characters like "果" and "裁" which have the components attached between two components. However, our method skips the segmentation process, we utilize the recombination of strokes to form the most similar set to form certain component. Sequently, we use the optimal combined strategy to select the best combination result, which at the same time, the components in character are segmented.

V. CONCLUSION

In this paper, a systematic statistical structure modeling method is improved. Combined with utilization of optimal combined strategy, we could deal with the problem of component recognition in Chinese characters. Both the characters and the components in image are represented by a set of strokes. Through matching strokes to instances using structure information, we could find the best matching result for every model component. Then solving the variant knapsack problem by optimal combined strategy, we could find the best combination of components for the input character. Then the components recognition results are extracted. Furthermore, the structural information could be achieved by structure model matching.

The experiment shows the method is acceptable in Chinese component extraction. But there are also problems. Some special characters like "森" and "品" demand us to search for more than one matching solution in the single matching stage, which makes the problem more complex. Also, the irregular character images should be put into experiment to show the robustness of our method. Optimizing the process of graph searching and improving experiment including more irregular character images will be our future work.

ACKNOWLEDGMENT

This work was supported by the Beijing Natural Science Foundation(Researches on Human Body Segmentation Methods in Natural Environment based on Computer Vision, Grant No.4112032), the Natural Science Foundation of China (Grant No.61170186).

REFERENCES

[1] Cheng Lin Liu and M.Nakagawa, *Prototype Learning Algorithms for Nearest Neighbor Classifier with Application to Handwritten Character Recognition*. Pattern Recgnition, 2001.

[2] Y.Y. Tang, L. T. Tu, J.Liu, S. W. Lee, and W. W. Lin, *Offline recognition of Chinese Handwriting by Multifeature and Multi-level Classification*. IEEE Trans.Pattern Analysis and Machine Intelligence, 1998.

[3] D.Shi, S.R. Gunn, and R.I. Damper, *Handwritten Chinese Radical Recognition Using Nonlinear Active Shape Models*. IEEE Trans.Pattern Analysis and Machine Intelligence, 2003.

[4] P. K. Wong and C. Chan, *Off-Line Handwritten Chinese Character Recognition as a Compound Bayes Decision Problem*. IEEE Trans.Pattern Analysis and Machine Intelligence, 1998.

[5] N.Kato, M.Suzuki, S.Omachi, H.Aso and Y.Nemoto, *A Handwritten Chinese Character Recognition System Using Directional Element Feature and Asymmetric Mahalanobis Distance*. IEEE Trans. Pattern Analysis and Machine Intelligence, 1999.

[6] H.J. Lee, B. Chen, *Recognition of handwritten Chinese characters via short segments*. Pattern Recognition, 1992.

[7] Cheng Lin Liu, In Jung Kim and Jin H. Kim, *Model-Based Stroke Extraction and Matching for Handwritten Chinese Character Recognition*. Pattern Recognition, 2001.

[8] D. Shi, S.R. Gunn and R.I. Damper, *Offline Handwritten Chinese Character Recognition by Radical Decomposition*. ACM Transactions on Asian Language Information Processing, 2003.

[9] Yanwei Wang, Xiaoqing Ding and Changsong Liu, *MQDF Discriminative Learning Based Offline Handwritten Chinese Character Recognition*. International Conference on Document Analysis and Recognition, 2011.

[10] In Jung Kim and Jin Hyung Kim, *Statistical Character Structure Modeling and Its Application to Handwritten Chinese Character Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003.

[11] Jia Zeng, Wei Feng,Lei Xie and Zhi Qiang Liu, *Cascade Markov Random Fields for Stroke Extraction of Chinese Character*. Information Sciences, 2010.

[12] Yih Ming Su, Jhing Fa Wang, *A Novel Stroke Extraction Method for Chinese Characters using Gabor Filters*. Pattern Recognition, 2003.

[13] A.B. Wang and K.C. Fan, *Optical Recognition of Handwritten Chinese Characters by Hierarchical Radical Matching Method*. Pattern Recognition, 2001.

Visually and Phonologically Similar Characters in Incorrect Chinese Words: Analyses, Identification, and Applications

C.-L. LIU, M.-H. LAI, K.-W. TIEN, and Y.-H. CHUANG, National Chengchi University
S.-H. WU, Chaoyang University of Technology
C.-Y. LEE, Academia Sinica

Information about students' mistakes opens a window to an understanding of their learning processes, and helps us design effective course work to help students avoid replication of the same errors. Learning from mistakes is important not just in human learning activities; it is also a crucial ingredient in techniques for the developments of student models. In this article, we report findings of our study on 4,100 erroneous Chinese words. Seventy-six percent of these errors were related to the phonological similarity between the correct and the incorrect characters, 46% were due to visual similarity, and 29% involved both factors. We propose a computing algorithm that aims at replication of incorrect Chinese words. The algorithm extends the principles of decomposing Chinese characters with the Cangjie codes to judge the visual similarity between Chinese characters. The algorithm also employs empirical rules to determine the degree of similarity between Chinese phonemes. To show its effectiveness, we ran the algorithm to select and rank a list of about 100 candidate characters, from more than 5,100 characters, for the incorrectly written character in each of the 4,100 errors. We inspected whether the incorrect character was indeed included in the candidate list and analyzed whether the incorrect character was ranked at the top of the candidate list. Experimental results show that our algorithm captured 97% of incorrect characters for the 4,100 errors, when the average length of the candidate lists was 104. Further analyses showed that the incorrect characters ranked among the top 10 candidates in 89% of the phonologically similar errors and in 80% of the visually similar errors.

Categories and Subject Descriptors: I.2.7 [**Computing Methodologies**]: Artificial Intelligence—*Natural language processing*; J.5 [**Computer Applications**]: Arts and Humanities—*Linguistics*; K.3.1 [**Computing Milieux**]: Computers and Education—*Computer uses in education; Computer-assisted instruction (CAI)*; H.3.5 [**Information Systems**]: Information Storage and Retrieval—*Online information services; Web-based services*; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Psychology*

General Terms: Design, Languages

Additional Key Words and Phrases: Error analysis of written Chinese text, student modeling, traditional Chinese, simplified Chinese, computer-assisted language learning, psycholinguistics

ACM Reference Format:
Liu, C.-L., Lai, M.-H., Tien, K.-W., Chuang, Y.-H., Wu, S.-H., and Lee, C.-Y. 2010. Visually and phonologically similar characters in incorrect Chinese words: Analyses, identification, and applications. ACM Trans. Asian Lang. Inform. Process. 10, 2, Article 10 (June 2011), 39 pages.
DOI = 10.1145/1967293.1967297 http://doi.acm.org/10.1145/1967293.1967297

This article was completed while C.-L. Liu visited the Department of Electrical Engineering and Computer Science of the University of Michigan as a visiting scholar.
This research was supported in part by the research contracts 97-2221-E-004-007, 99-2221-E-004-007, and 99-2918-I-004-008 from the National Science Council of Taiwan.
Authors' addresses: C.-L. Liu, M.-H. Lai, K.-W. Tien, and Y.-H. Chuang, Department of Computer Science, College of Science, National Chengchi University, Taipei, Taiwan; email: {chaolin, g9523, g9627, g9804}@cs.nccu.edu.tw; S.-H. Wu, Department of Computer Science and Information Engineering, College of Informatics, Chaoyang University of Technology, Taichung, Taiwan; email: shwu@cyut.edu.tw; C.-Y. Lee, Institute of Linguistics, Academia Sinica, Taipei, Taiwan; email: chiaying@gate.sinica.edu.tw.

DOI 10.1145/1967293.1967297 http://doi.acm.org/10.1145/1967293.1967297

1. INTRODUCTION[1]

The studies about people using incorrect characters in Chinese words are related to the education, perception, recognition, and applications of the Chinese language[2]. Some Chinese words contain just one character, but most words comprise two or more characters. For instance, "好" (hao3)[3] is a word that has just one character and means "good" in English. "語言" (yu3 yan2) is a word that is formed by two characters and means "language" in English. Experience indicates that the two most common causes for writing or typing incorrect Chinese words are due to phonological and visual similarity between the correct and the incorrect characters [Liu et al. 2009a, 2009b, 2009c]. For instance, one might use "素" (su4) in the place of "肅" (su4) in "嚴肅" (yan2 su4) because of the phonological similarity; one might use "施" (shi1) for "旅" (lu3) in "旅途" (lu3 tu2) due to the visual similarity.

Manipulating the similarity between characters has served as an instrumental technique in psycholinguistic studies into how people read and recognize Chinese characters. Researchers in psycholinguistics investigate the cognition processes of Chinese readers [Kuo et al. 2004; Lee et al. 2006; Tsai et al. 2006], by measuring readers' response times to words that have various numbers of "neighbor" words. The neighbors of a Chinese word include phonologically and visually similar characters.

Phonologically and visually similar characters are also useful for computer assisted language learning (CALL). In elementary schools in Taiwan, students may be requested to identify and correct "erroneous words" in test items, where, typically, an "erroneous word" contains an incorrect character that was introduced intentionally when teachers prepared the test items. Such tests are Incorrect Character Correction tests (ICC tests). It takes effort and time to provide incorrect characters that are appropriate for different assessment purposes, and to make sure that the test items do not repeatedly use the same incorrect characters at the same time. We have built an environment for assisting the preparation of such test items [Liu et al. 2009a] by finding a way to offer phonologically and visually similar Chinese characters as candidates to serve as the incorrect characters [Liu and Lin 2008].

In addition, phonologically and visually similar characters can be applied to student modeling, optical character recognition (OCR), and information retrieval (IR) in Chinese. Bug libraries contain students' records of previous errors [Sison and Shimura 1998; Virvou et al. 2000], and are useful for modeling student behavior. Some algorithms for optical character recognition for printed Chinese and for written Chinese try to guess the input images based on confusion sets [Fan et al. 1995; Liu et al. 2004]. Characters in a confusion set are similar to each other visually, and they help the OCR programs to confine the search space for a given image. It would be possible to reduce the computational costs and to increase recognition rates if we can pinpoint the confusion set of a character that is being recognized. The current confusion sets are hand-crafted clusters of visually similar characters. In recent years, it has become a

[1]This article is a significantly extended version, in terms of the depth of discussion and the scale of experiments, of the material reported in Liu and Lin [2008], Liu et al. [2009a, 2009b, 2009c], and Liu et al. [2010].

[2]In this article, we use "Chinese" to refer to Mandarin Chinese.

[3]We show traditional and simplified Chinese characters followed by their Hanyu pinyin (http://en.wikipedia.org/wiki/Pinyin). The Hanyu pinyin of a Chinese character shows the sound of the character by a string of English letters, and the digit that follows the letters is the tone for the character. To simplify our presentation, we will show Chinese text only in either the traditional or the simplified form, but not both. If presented in simplified Chinese, the errors listed in the first paragraph in the Introduction will replace "肅" (su4) in "严肃" (yan2 su4) by "素" (su4) for phonological similarity and "旅" (lu3) in "旅途" (lu3 tu2) by "施" (shi1) for visual similarity. The traditional and simplified forms of a Chinese character might not differ from each other.

common practice for IR service providers, such as Yahoo! and Google, to offer corrections when users enter queries that contain incorrect words. For English queries, one may apply the Levenshtein distance to compute the edit distance between the spellings and employ the Soundex system to determine the degree of similarity between the pronunciations of words [cf., Croft et al. 2010; Manning et al. 2008]. These methods are not perfect but can catch similar English words in practice. The work reported in this article can be applied to find possible corrections for Chinese queries.

Some researchers state that there are more than 50,000 Chinese characters [HanDict 2010], although only thousands of characters are used in daily lives. In the People's Republic of China, a government agency selected 7,000 popular Chinese characters and highlighted 3,500 characters among these 7,000 characters as the most frequently used characters in 1988[4]. In Taiwan, 5,401 characters were selected to be the most commonly used in daily lives in 1984 when the BIG5 code was formulated [Dict 2010].

Given that Chinese is used in different areas and in different countries in the world, it should not be surprising that not all people speak the "standard" Mandarin. We will focus on the standards that are stated in specific lexicons in this research. Given a specific lexicon, it is relatively easy to judge whether two characters have the same or similar pronunciations based on their records, when we do not consider the phenomena of co-articulation. Although there are thousands of Chinese characters, these characters are pronounced in only 420 different ways (cf., Lee [2009]). Interestingly, there are many fewer Chinese words that are pronounced exactly the same way than the number of Chinese characters that are pronounced exactly the same way. The problem of determining the pronunciation of Chinese characters becomes more complex if we consider tone sandhi [Chen 2000] in Chinese and if we consider the influences of sublanguages in the Chinese language family. We will discuss related issues in Section 2.

In contrast, there were no obvious ways to determine algorithmically whether two Chinese characters are visually similar yet. For instance, "員" (yuan2), "圓" (yuan2), and "勛" (xun1) are similar to each other in some ways, due to the presence of "員" (yuan2). Image processing techniques may be useful but are not perfectly practical, given the size of Chinese characters. A more important factor that affects the applicability of image processing methods is that many of the Chinese characters are similar to each other in subtle ways. "員" (yuan2) is contained in "員" (yuan2), "圓" (yuan2), and "勛" (xun1) in different sizes and at different positions.

We apply an extended version of the Cangjie codes [Cangjie 2010; Chu et al. 2010] to encode the layouts and details of traditional Chinese characters for computing visually similar characters [Liu and Lin 2008; Liu et al. 2009a, 2009b, 2009c], and extend the work to compare similar characters in simplified Chinese characters [Liu 2010]. Evidence observed in psycholinguistic studies [Feldman and Siok 1999; Lee et al. 2006; Yeh and Li 2002] offers a cognition-based support for the design of our approach; namely, the use of shared components to define the visual similarity between Chinese characters.

The proposed method proves to be effective in capturing incorrect words for both traditional [Liu et al. 2009a, 2009b, 2009c] and simplified Chinese [Liu 2010]. We col-

[4]The statistics are available on the following two Wikipedia pages: http://zh.wikipedia.org/zh-tw/%E7%8E%B0%E4%BB%A3%E6%B1%89%E8%AF%AD%E9%80%9A%E7%94%A8%E5%AD%97%E8%A1%A8 (if Chinese is available on your computers: http://zh.wikipedia.org/zh/现代汉语通用字表) and http://en.wikipedia.org/wiki/Xi%C3%A0nd%C3%A0i_H%C3%A0ny%C7%94_Ch%C3%A1ngy%C3%B2ng_Z%C3%ACbi%C7%8Eo (if Chinese is available on your computers: http://zh.wikipedia.org/zh/现代汉语常用字表). The first page is written in Chinese, and the second one is in English. The translations of "现代汉语" (xian4 dai4 han4 yu3) and "字表" (zi4 biao3) are "Modern Chinese" and "character list", respectively. We use "popular" for "通用" (tong1 yong4) and "most frequently used" for "常用" (chang2 yong4).

lected and analyzed approximately 4,100 errors that were reported in published books, found in students' compositions, or posted on the Internet. Each reported error is of a word which will be understood as appearing in its correct form as "嚴肅"; but which in the error may appear as "嚴素", where "素" is used instead of "肅". Namely, writing "嚴肅" as "嚴素" is a reported error. We found that 76% of the errors were related to phonological similarity and that 46% of the errors were related to visual similarity. More significantly, the dominance of the phonological factor was also observed in hand-written text, not just in electronic documents that were directly prepared on computers.

In experiments that aimed at reproducing the collected errors, we ran our programs to select and recommend a list of candidates from more than 5,100 Chinese characters for the correct character, that is, "肅", and we recorded the likelihood that the candidate list actually included the incorrect character. Experimental results show that if the length of the candidate list is about 100, we achieved inclusion rates of about 97% for both traditional and simplified Chinese. If the length of the candidate list was shortened to 10, the average inclusion rates were 89% for the phonologically similar errors and 80% for the visually similar errors. We have also applied our algorithms for reproducing the reported errors to build an environment to assist teachers to prepare test items for ICC tests.

In this article, we integrate and extend the previous reports on the phonologically and visually similar characters in both traditional and simplified Chinese to capture errors in Chinese words. We go over some issues about phonological similarity in Chinese in Section 2, elaborate how we extend and apply the Cangjie codes to judge the visual similarity between Chinese characters in Section 3, explain how we acquired the reported errors and how we analyzed the phonological and visual influences on these errors in Section 4, present details about our experiments and discuss the observations in Section 5, show a real-world application of the proposed techniques to the authoring of test items for the ICC tests in Section 6, and review some of the design issues and experience in Section 7 before we summarize our work in Section 8.

Compared with the previous conference articles [Liu and Lin 2008; Liu et al. 2009a, 2009b, 2009c; Liu et al. 2010], we expanded the scale of experiments and discussions in terms of both depth and coverage. More specifically, we validated the reliability of the Web-based statistics by examining the data that we collected in 2009 and in 2010, compared the contribution of different sources of similar characters, explored the applications of alternative ranking methods, and exhibited the robustness of our approach by running our systems over new data sets.

2. PHONOLOGICALLY SIMILAR CHARACTERS

Chinese characters are single syllable. The pronunciation of a Chinese character involves the nucleus and a tone, where the nucleus contains a vowel that follows an optional consonant. In this article, we use the Hanyu pinyin method to denote the sound of Chinese characters, and show the tone with a digit that follows the symbol string for the sound. In Mandarin Chinese, there are four tones. (Some researchers include the fifth tone.)

Although Chinese is not an alphabetical language, it is shown that the pronunciations of characters affect how people write Chinese [Ziegler et al. 2000]]. The pronunciation of a Chinese character has two parts: sound and tone. Therefore, the phonological similarity between two characters may consider these two aspects, and we consider four categories of phonological similarity between two characters: same sound and same tone (SS), same sound and different tone (SD), similar sound and same tone (MS), and similar sound and different tone (MD).

Table I. Samples of the Similar Phonemes with Example Characters

	Original Phoneme	Similar Phoneme	A Character with the Original Phoneme	Examples with the Similar Sound and the Same Tone (MS)
consonant	/s/	/sh/	肅 (**s**u4)	數、樹、恕 (**sh**u4)
	/z/	/zh/	早 (**z**ao3)	找、沼、爪 (**zh**ao3)
	/c/	/ch/	從 (**c**ong2)	重、虫、崇 (**ch**ong2)
vowel	/eng/	/en/	徵 (zh**eng**1)	真、針、貞 (zh**en**1)
	/eng/	/ang/	徵 (zh**eng**1)	張、章、蟑 (zh**ang**1)

We rely on the information provided in a lexicon [Dict 2010] to determine whether two characters have the same sound or the same tone. The judgment of whether two characters have a similar sound should consider the language experience of an individual. An individual who lives in southern China and one who lives in northern China, for instance, might have quite different perceptions of similar sound. In this work, we resort to the confusion sets observed in a psycholinguistic study, conducted at the Academic Sinica in Taiwan, to obtain a list of confusion sets of vowels and consonants in Mandarin Chinese.

Some Chinese characters are heteronyms [cf., Fromkin et al. 2002]. Let C_1 and C_2 be two characters that have multiple pronunciations. If C_1 and C_2 share one of their pronunciations, we consider that C_1 and C_2 belong to the SS category. This principle applies when we consider phonological similarity in other categories.

With a lexicon and the list of confusion sets, our program can select a list of phonologically similar characters for a given character. Consider the example "嚴肅" (yan2 su4) that we mentioned in Section 1. We can find all of the characters that have exactly the same pronunciation with "肅" (su4) based on the information provided by a lexicon. The SS list of "肅" will include characters such as "素" (su4) and "速" (su4). It is also easy to find the SD list for "肅", and it includes "蘇" (su1), "俗" (su2), and other characters. Based on the results of the psycholinguistic studies, we know that one might confuse the consonant /s/ with the consonant /sh/. Hence, the characters "數" (**sh**u4) and "樹" (**sh**u4) are in the MS list for "肅" (**s**u4), and the characters "書" (**sh**u1), "數" (**sh**u3), and "暑" (**sh**u3) are in the MD list for "肅". Notice that the character "數" is in the MS and MD lists for "肅" because it is a heteronym.

Table I shows more pairs of the confusing phonemes that we used in our system. Note that phonological similarity is a symmetric relationship. Namely, when phoneme X is similar to phoneme Y, phoneme Y is similar to phoneme X. To help readers focus on the symbols of the phonemes, we underline the confusing phonemes of the example characters in boldface. Notice also that, although we do not explicitly provide examples in Table I, it is possible to change both the consonant and the vowel for a character to find a phonologically similar character. For instance, "髒" (**z**ang1) is phonologically similar to "徵" (**zh**eng1) because we can replace the consonant /zh/ and vowel /eng/ in "徵" with /z/ and /ang/, respectively, and find "髒".

One challenge in defining phonological similarity between characters is that a Chinese character may be pronounced in more than one way, and the actual pronunciation depends on the context. Tone sandhi [Chen 2000] is a frequently mentioned source of confusion. The most common example of the use of tone sandhi in Chinese is that the first third-tone character in words formed by two adjacent third-tone characters will be pronounced with the second tone. For example, although "你" (ni3) and "好" (hao3) are both third-tone characters, "你" in "你好" is pronounced with the second

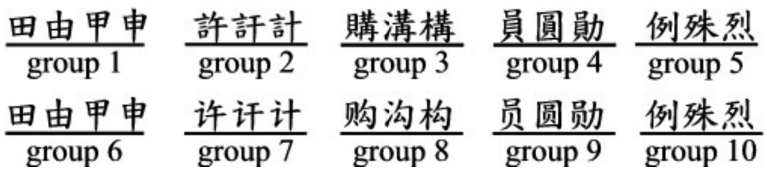


Fig. 1. Examples of visually similar characters in traditional Chinese (groups 1-5) and in simplified Chinese (groups 6-10).

tone in practice. Namely, native speakers usually pronounced "你好" as ni2-hao3. At present, we ignore the influences of context when determining whether two characters are phonologically similar. (As we shall see in Section 5, doing so did not disturb the experimental results.)

Although we have confined our definition of phonological similarity to the context of the Mandarin Chinese, we would like to note that the influence of sublanguages within the Chinese language family will affect the perception of phonological similarity. Dialects used in different areas in China, for example, Shanghai, Min, and Canton, share the same written forms with the Mandarin Chinese, but have quite different though related pronunciation systems. Hence, people living in different areas in China might perceive phonological similarity in different ways. The study in this direction, however, is beyond the scope of the current study.

3. VISUALLY SIMILAR CHARACTERS

Figure 1 shows examples of visually similar Chinese characters. The first row contains five groups of visually similar traditional Chinese characters, and the second row contains five corresponding groups of simplified Chinese characters. The j^{th} character (counted from left to right) in group $(i + 5)$ is the simplified form of the j^{th} character in group i. Notice that the traditional and simplified forms of a character may be exactly the same.

The characters in group 1 differ subtly at the stroke level, as do the characters in group 2. The characters in group 3 share the same components on their right sides. The shared components of the characters in group 4 and group 5 appear at different places within the characters.

Analogously, characters in group 6 differ subtly at the stoke level, as do the simplified characters in group 7. Characters in group 8 share the components on their right sides. The shared components of the characters in group 9 and group 10 appear at different places within the characters.

The radical of a Chinese character carries the main semantic information about the character [cf., Feldman et al. 1999], and lexicographers employ radicals to organize characters in Chinese dictionaries. Characters that belong to the same radicals are placed in the same category, and are listed sequentially by the number of strokes. Hence, it is possible to employ the information about radicals to find visually similar characters. The characters in group 1 and group 2 have the radicals "田" (tian2) and "言" (yan2), respectively. Analogously, the simplified characters in group 6 and group 7 have the radicals "田" and "讠", respectively. ("讠" is the simplified form of "言".) Notice that, although the radicals for group 2 and group 7 are obvious, those for group 1 and group 6 are not because "田" is not a standalone component in these groups.

Although radicals themselves provide information about the shared components of characters, the most saliently shared components of characters might not be the radicals of the characters. This problem occurs in both traditional and simplified Chinese. The shared component of the characters in group 3 is not the radical. The

shared components of the characters in groups 4, 8, and 9 are not the radicals for the characters in the groups either. In these cases, the shared components carry information about the pronunciations of the characters. Hence, those characters are listed under different radicals, though they do look similar in some ways.

In some cases, one may be interested in characters that share small elements in the characters, such as "歹" (dai3) in group 5 and group 10. The shared elements in these two groups do not carry semantic or phonological information, and they are not the radicals either. It is also possible that a radical is written in different ways in the characters that have the same radical in a dictionary, for example, "泉" (quan2) and "泊" (bo2). These two characters are listed under the radical "水" (shui3). The radical appears literally in "泉", but is written as "氵" in "泊".

Therefore, we cannot rely only on the information about radicals of characters in typical lexicons to find visually similar characters, and we will use the extended Cangjie codes as the basis to judge the degree of similarity between Chinese characters.

3.1 Cangjie Codes for Traditional Chinese

The Cangjie input method is one of the most popular methods used for entering Chinese characters into computers. The designer of the Cangjie method selected a set of 24 basic elements occurring in characters, and proposed a set of rules to decompose Chinese characters according to these elements [Chu et al. 2010]. Because the Cangjie system is designed to help people enter Chinese characters into computers, the design of the Cangjie codes had aimed at allowing its users to recall the codes for Chinese characters as easy as possible. Namely, users must be able to easily figure out the Cangjie codes for the characters that they want to enter. Given the popularity of the Cangjije input method in a wide range of Chinese speaking communities, the Cangjie codes of Chinese characters have practically shown their strong links with the formation of Chinese characters. This was an important motivation for us to try to define the similarity between two Chinese characters based on the degree of similarity between their Cangjie codes.

Table II shows the Cangjie codes for the 13 characters listed in groups 1 to 4 in Figure 1 and for five other characters. The "ID" column shows the identification number for the characters, and we will refer to the i^{th} character by c_i, where i is the ID. The "CC" column shows the Chinese characters, and the "Cangjie" column shows the Cangjie codes. Each symbol in the Cangjie codes corresponds to a key on the keyboard, for example, "田" (tian2) and "中" (zhong1) collocate with "W" and "L", respectively. Information about the complete correspondence is available in Wikipedia[5].

Using the Cangjie codes saves us from the need to apply image processing methods to determine the degrees of similarity between characters. Take the Cangjie codes for the characters in group 2 (c_5, c_6, and c_7) for example. See that the characters share a common component based on the shared substrings of the Cangjie codes (shown in boldface), that is, "卜 ロ" (bu3 kou3). We may also find the shared component "冓" (gou4, encoded by "廿廿月") for the characters in group 3 (c_{10}, c_{11}, and c_{12}), the shared component "力" (li4, encoded by "大尸") in c_{15} and c_{16}, and the shared component "巫" (jing1, encoded by "一一") in c_{16} and c_{17}.

However, the original Cangjie codes are still lacking in some respects, in spite of their perceivable advantages. The Cangjie codes have been limited to contain no more than five keys, in order to maintain efficiency in inputting Chinese characters. Thus,

[5]See http://en.wikipedia.org/wiki/Cangjie_input_method#Keyboard_layout; last visited on 30 September 2010.

Table II. Examples of Cangjie Codes for Traditional Chinese

ID	CC	Cangjie	ID	CC	Cangjie
1	田	田	10	購	月金廿廿月
2	由	中田	11	溝	水廿廿月
3	甲	田中	12	構	木廿廿月
4	申	中田中	13	員	口月山金
5	許	卜口人十	14	圓	田口月金
6	訐	卜口一十	15	勛	口金大尸
7	計	卜口十	16	勁	二二大尸
8	政	二二人大	17	頸	二二一月金
9	元	二二山	18	經	女火一女一

Table III. Examples of Cangjie Codes for Simplified Chinese

ID	CC	Cangjie	ID	CC	Cangjie
19	田	田	28	购	月人心戈
20	由	中田	29	沟	水心戈
21	甲	田中	30	构	木心戈
22	申	中田中	31	员	口月人
23	许	戈女人十	32	圆	田口月人
24	讦	戈女一十	33	勋	口人大尸
25	计	戈女十	34	劲	弓一大尸
26	鲳	弓一日日	35	颈	弓一一月人
27	驹	弓一心口	36	经	女一弓人一

users of the Cangjie input method must familiarize themselves with the principles for simplifying the Cangjie codes. While the simplified codes help to enhance the input efficiency, they also introduce difficulties and ambiguities when we compare the original Cangjie codes for computing similar characters. The shared component "員" (yuan2) is encoded in three different ways in c_{13}, c_{14}, and c_{15}, i.e., "口月山金" (kao2 yue4 shan1 jin1), "口月金", and "口金". The prefix "一一" in c_{16} and c_{17} can represent "巫" (jing1), "正" (zheng4; e.g., in c_8), and "二" (er4; e.g., in c_9). Consequently, characters whose Cangjie codes include "一一" may contain any of these three components, but c_8, c_9, and c_{16}do not really look alike.

3.2 Cangjie Codes for Simplified Chinese

Not surprisingly, the Cangjie codes are also useful for capturing the similarities between simplified Chinese characters. Using a structure similar to Table I, Table III shows the Cangjie codes for the characters listed in groups 6 to 9 in Figure 1 and five other characters.

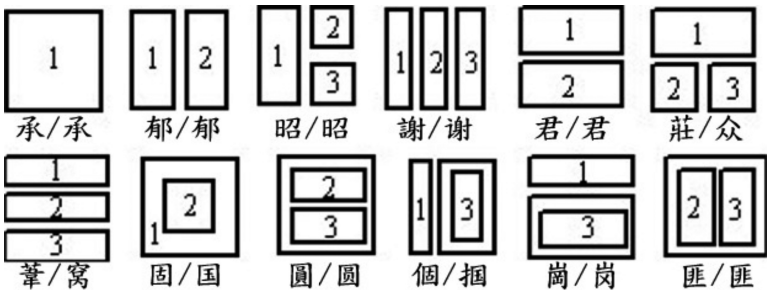


Fig. 2. Layouts of Chinese characters (used in Cangjie).

Again, the Cangjie codes offer the possibility to determine the degrees of similarity between characters efficiently. It is possible to find that the characters c_{23}, c_{24}, and c_{25}share a common component because their Cangjie codes share "戈女" (ge1 nu3). Using the common substrings (shown in boldface) of the Cangjie codes, we may also find the shared component "勾" (gou1, encoded by "心戈") for the characters in group 8 (c_{28}, c_{29}, and c_{30}), the shared component "员" (yuan2, encoded by "口月人") in c_{31} and c_{32}, the shared component "力" (li4, encoded by "大尸") in c_{33} and c_{34}, and the shared component "巠" (jing1, encoded by "弓一") in c_{34} and c_{35}.

Similar to the problem of using the original Cangjie codes for traditional Chinese, we would encounter ambiguity problems when comparing the similarities between simplified Chinese characters. The shared component "员" in c_{32} and c_{33} is encoded by "口月人" (kao3 yue4 ren2) and "口人", respectively. The prefix "弓一" (gong1 yi1) in c_{34} and c_{35} can represent "巠", "鱼" (yu2; e.g., in c_{26}), and "马" (ma3; e.g., in c_{27}). Characters whose Cangjie codes include "弓一" may contain any of these three components, but c_{26}, c_{27},and c_{34} do not really look alike.

Given the observations reported in the previous subsection and in this present one, we augmented the original Cangjie codes by using the complete Cangjie codes and annotated each Chinese character with a layout identification that encodes the overall contours of the characters.

3.3 Augmenting the Cangjie Codes

Figure 2 shows the 12 possible layouts that are considered for the Cangjie codes for both traditional and simplified Chinese characters. Most of the layouts contain two or three small regions (called subareas henceforth), and the rectangles show individual subareas within a character. The subareas are assigned IDs, but to maintain readability of the figures, not all of the IDs for subareas are shown in Figure 2. From left to right and from top to bottom, each layout is assigned an identification number from 1 to 12. An example pair of characters, separated by a slash, is provided below each layout. A traditional Chinese character is on the left, and a simplified one is on the right. For example, the layout ID of "固/国" is 8. "固" (gu4) is a traditional Chinese character, and has two parts, that is, "口" (wei2) and "古" (gu3). "国" (guo2) is a simplified Chinese character and has two parts, that is, "口" and "玉" (yu4).

When Chinese characters are transformed from the traditional to simplified forms, the layout of the same characters may or may not be changed, and a more comprehensive discussion about the significant change in the structures of the characters is available in Lee [2010b]. Hence, we may and may not use the traditional and simplified forms of the same character as a pair in Figure 2. Except for layouts 6, 7, 8, and 10, the pairs of characters shown under the layouts are the same characters in both traditional and simplified forms. For instance, "谢" (xie4) and "謝" (xie4) are

examples of layout 4, and "谢" is the simplified form of "謝". In contrast, "囿" and "国" are two different characters, but both are examples of layout 8. The traditional form of "国" is "國", which belongs to layout 9.

Researchers have come up with other ways to decompose individual Chinese characters. A team at the Shanghai Jiao-Tong University (SJTU) report an early attempt, and they consider five major ways to decompose Chinese characters [p. 1071, SJTUD 1988]. In this study, the SJTU team report detailed analysis of the compositions of Chinese characters. Based on their analysis, "口" (kou3) and "木" (mu4) are the most frequent components in Chinese characters [p. 1027, SJTUD 1988]. Juang et al. [2005] employ four relationships for components of Chinese characters, and Sun et al. [2002] six relationships. The Chinese Document Laboratory at the Academia Sinica in Taiwan considers 13 possible ways to decompose Chinese characters [CDL 2010]. Lee [2010b] proposes more than 30 possible layouts. In Unicode standard 4.0.1, 12 operators are considered to build Chinese characters from a set of building blocks [UNICODE 2010].

The layout of a character affects how people perceive the visual similarity between characters [Yeh and Li 2002]. For instance, c_{16} ("勁", jing4) in Table II is more similar to c_{17} ("頸", jing3) than to c_{18} ("經", jing1), because the shared component of c_{16} and c_{17} is on the same side of the words. Overall, c_{16}, c_{17}, and c_{18} are more similar to each other than to "瘜" (jing1), although they share "巠" (jing1). We follow the style by which Chinese characters are decomposed in the Cangjie system, and rely on the expertise in Cangjie codes reported in Lee [2010a] to divide a Chinese character into subareas, which we showed in Figure 2.

Table IV shows the extended codes for some of the characters listed in Table I. The ID column provides links between the characters listed in both Table II and Table IV. The CC column repeats the Chinese characters. The LID column shows the identifications for the layouts of the characters. The columns with headings P1, P2, and P3 show parts of the extended Cangjie codes, where Pi shows the i^{th} part of the Cangjie codes, as indicated in Figure 2.

LIDs are useful for comparing the degree of similarity between characters. Consider the case that we want to determine whether "勁" (jing4) is more similar to "頸" (jing3) than it is similar to "經" (jing1). Their extended Cangjie codes indicate that "頸" is a better answer for two reasons. First, both "勁" and "頸" are examples of layout 2; and, second, the shared components reside in the same subarea, that is, P1, in "勁" and "頸".

We decide the extended Cangjie codes for the individual parts with the help of computer programs and subjective judgments. Starting from the original Cangjie codes, we can compute the most frequent substrings among the original Cangjie codes for all characters. This process is similar to the one which can be used to compute the frequencies of n-grams in corpora [cf., Jurafsky and Martin 2009]. Computing the most frequently appearing substrings in the original codes is not a complex task because the longest original Cangjie codes contain just five symbols.

Often, the frequent substrings are simplified codes for common components in Chinese characters, for example, "言" (yan2) and "巠" (jing1). The complete codes for "言" and "巠" should be "卜一一口" (bu3 yi1 yi1 kou3) and "一女女女一" (yi1 nu2 nu2 nu2 yi1), but they are simplified to "卜口" and "一一", respectively, in the original Cangjie codes. When the Cangjie codes are simplified, "巠" has the same code as "正" (zheng4) and "二" (er4), as we have illustrated by c8, c9, and c16 in Table II. The simplified Cangjie codes for "言" are the same as the Cangjie codes of "亠", which is in the upper part of "高" (gao1).

After finding the frequent substrings, we verify whether these frequent substrings are simplified codes for meaningful components, which, in our definition, form parts

Table IV. Examples of Extended Cangjie Codes for Traditional Chinese

ID	CC	LID	P1	P2	P3
5	許	2	卜**一一**口	人十	
6	訐	2	卜**一一**口	一十	
7	計	2	卜**一一**口	十	
10	購	10	月**山**金	廿廿	**土**月
11	溝	10	水	廿廿	**土**月
12	構	10	木	廿廿	**土**月
13	員	5	口	月山金	
14	圓	9	田	口	月**山**金
15	勛	2	口**月山**金	大尸	
16	勁	2	一**女女女**一	大尸	
17	頸	2	一**女女女**一	一月**山**金	
18	經	3	女**戋**火	一女**女女**	一

of one or more Chinese characters. For meaningful components, we replace the simplified codes with their complete codes. For instance, the Cangjie codes for "許" (xu3) and "訐" (jie2) are extended to contain "卜一一口" in Table IV, where we indicate the extended keys that did not belong to the original Cangjie codes in boldface and with a surrounding box. After recovering the dropped codes for "言", our programs will have the information necessary to be able to tell "言" and "古" apart.

Although we have tried to employ computer programs to help us find the frequent substrings in as many instances as we can, the work to recover the simplified codes remained labor-intensive, and we had to devote particular attention to certain anomalous cases at times. Fortunately, the process to implement the extended Cangjie codes proved to be worthwhile as we will show in the experimental studies.

Using a structure that is similar to Table IV, Table V shows the extended Cangjie codes for some of the simplified Chinese characters that we show in Table III. The "ID" column provides links between the characters listed in both Table III and Table V.

In Table V, we recover the Cangjie codes for "讠" (yan2) and "圣" (jing1). Using "戈弓女" (ge1 gong1 su3), rather than "戈女", for "讠" prevents us from confusing "讠" with "戉" (yue4). Similarly, using "弓人一" (gong1 ren2 yi1), rather than "弓一", for "圣" avoids the confusion of "马" (ma3) and "鱼" (yu2) with "圣", e.g., c26, c27, and c34 in Table III.

Replacing simplified codes with complete codes not only helps us avoid incorrect matches but also helps us find matches that would be missed due to the simplification of the Cangjie codes. If we only use the original Cangjie codes in Table III, it is not easy

Table V. Examples of Extended Cangjie Codes for Simplified Chinese

ID	CC	LID	P1	P2	P3
23	许	2	戈弓女	人十	
24	讦	2	戈弓女	一十	
25	计	2	戈弓女	十	
28	购	10	月人	心	戈
29	沟	10	水	心	戈
30	构	10	木	心	戈
31	员	5	口	月人	
32	圆	9	田	口	月人
33	勋	2	口月人	大尸	
34	劲	2	弓人一	大尸	
35	颈	2	弓人一	一月人	
36	经	3	女女一	弓人	一
37	恸	4	心	一一戈	大尸

to determine that c36 ("经", jing1) in Table III shares the component "圣"(jing1) with c34 ("劲", jing4) and c35 ("颈", jing3). In contrast, there is a chance to find the similarity with the extended Cangjie codes in Table V, given that all of the three Cangjie codes include "弓人一" (gong1 ren2 yi1).

Although most of the examples provided in Table V indicate that we expanded only the first part of the Cangjie codes for the simplified Chinese, it is possible that the other parts, that is, P2 and P3, may need to be extended too. Sample c37 shows such an example.

3.4 Similarity Measure

The main differences between the original and the extended Cangjie codes are the degrees of detail about the structures of the Chinese characters. By recovering the details that were ignored in the original codes, our programs will be better equipped to find the similarities between characters.

We experiment with three different scoring methods to measure the visual similarity between two characters based on their extended Cangjie codes. Two of these methods were tried in our studies for traditional Chinese characters [Liu et al. 2009b, 2009c]]. The first method, denoted SC1, considers the total number of matched keys in the matched parts. Two parts are considered as matched as long as their contents are the same. They do not have to locate at the region within a character. Let c_i denote the i^{th} character listed in Table V. We have SC1(c_{33}, c_{34}) = 2 because of the matched "大尸"

(da4 shi1). Analogously, we have SC1(c_{37}, c_{34}) = 2 because of the matched "大尸". Notice that, although "一" (yi1) is in the P1 of c_{34} and in the P2 of c_{37}, it is not considered a match because the P1 of c_{34} and the P2 of c_{37} do not match as a whole.

The second method, denoted SC2, includes the score of SC1 and considers the following conditions: (1) add one point if the matched parts locate at the same place in the characters and (2) if the first condition is met, an extra point will be added if the characters belong to the same layout. Hence, we have SC2(c_{33}, c_{34}) = SC1(c_{33}, c_{34}) + 1 + 1 = 4 because (1) the matched "大尸" is the P2 of both characters and (2) c_{33} and c_{34} belong to the same layout. Assuming that c_{34} belongs to layout 5, than SC2(c_{33}, c_{34}) would become 3. In contrast, we have SC2(c_{37}, c_{34}) = 2. No extra points were added for "大尸" in the Cangjie codes for c_{37} and c_{34} because "大尸" is not at the same position in the characters. The extra points consider the spatial influences of the matched parts on the perception of similarity.

While splitting the extended Cangjie codes into parts allows us to tell that c_{33} is more similar to c_{34} than to c_{37}, it also creates a new barrier in computing similarity scores. An example of this problem is that SC2(c_{35}, c_{36}) = 0. This is because that "弓人一" (gong1 ren2 yi1) at P1 in c_{35} can match neither "弓人" at P2 nor "一" at P3 in c_{36}.

To alleviate this problem, we consider SC3 which computes the similarity in three steps. First, we concatenate the parts of a Cangjie code for a character. Then, we compute the longest common subsequence (LCS) (cf., Cormen et al. [2009]) of the concatenated codes of the two characters being compared, and compute a Dice coefficient (cf., Croft et al. [2010]) as the similarity. The Dice coefficients are used in many applications, including defining the strength of the relatedness of two terms (or similarity of two strings) in natural language processing (cf., Manning and Schütze [1999]). Let X and Y denote the concatenated, extended Cangjie codes for two characters, and let Z be the LCS of X and Y. The similarity is defined by the following equation.

$$Dice_{LCS} = \frac{2 \times |Z|}{|X| + |Y|}, \text{ where } |S| \text{ is the length of string } S \tag{1}$$

We compute another Dice coefficient between X and Y. The formula is the similar to Equation (1), except that we set Z to the longest common consecutive subsequence. We call this score $Dice_{LCCS}$. Notice that $Dice_{LCCS} \leq Dice_{LCS}$, $Dice_{LCCS} \leq 1$, and $Dice_{LCS} \leq 1$. Using both $Dice_{LCS}$ and $Dice_{LCCS}$ allows us to compute the visual similarity from two aspects. Finally, the SC3 of two characters is the sum of their SC2, $10 \times Dice_{LCCS}$, and $5 \times Dice_{LCS}$. We multiply the Dice coefficients with constants to make them as influential as the SC2 component in SC3. Since the LCCSs of two strings are generally quite shorter than the LCSs, we multiply $Dice_{LCCS}$with a larger weight. The constants were not scientifically chosen, but were selected heuristically.

Using the extended Cangjie codes and a selected score function, we can select a list of visually similar characters for a given character. Using SC3, we can find that every character in the string "逕涇輕徑痙莖紜氫" (jing4 jing1 qing1 jing4 jing4 jing1 yun2 qing1) is similar to "經" (jing1) in some way. Interestingly, each character in the string "逕涇輕徑痙莖紜氫" belongs to different radicals: "辶氵車彳疒艸糸气" (chuo4 shui3 che1 chi4 chuang2 cao3 mi4 qi4). One may verify that not all of the characters in the string are listed under the same radical as "經", so our approach offers chances to find visually similar characters that belong to different radicals.

3.5 Appropriate Similarity Measures

We discuss the main functions of these measures qualitatively, before we compare their effectiveness experimentally in Section 5.

Consider the problem of finding a set of "visually similar characters" for a given character within a Chinese word. Except resorting to the remembrance of human experts, how can we find similar characters from thousands of characters? A more important question may be what we mean by "similar" characters. Certainly, some characters can look similar individually. However, when putting into the contexts of words, some words can become more attractive than others. For instance, "浩" (hao4), "皓" (hao4), and "治" (zhi4) are almost equally similar to each other in some ways. Each pair differs in only in one of their components. Replacing the radical "氵" (shui3) in "浩" with the radical "白" (bai2) will create "皓", and replacing the component "告" (gao4) in "浩" with the component "台" (tai2) will create "治". However, "皓大" (hao4 da4), and "治大" (zhi4 da4) are not equally attractive for the writing of "浩大" (hao4 da4). Psycholinguistic evidence has shown that humans do not read text letter by letter for alphabetic languages or character by character for languages such as Chinese (e.g., Jackendoff [1995]). The contexts matter in determining the similarities.

As a result, the "best" similarity measure for computer software depend on the goals of the applications. Do we want to build a model for how humans judge visually-similar characters? Do we want to build a model for how human process confusing words in which some characters are visually similar?

In this article, the target application is more closely related to the latter question. In another application that we are building for learning Chinese characters [Liu et al. 2011], we are more concerned with similarity between individual characters. Hence, the similarity measures that we presented in the previous section were just to find "good candidates", and we will have another measure to compare these first-round candidates.

This is the main reason that we did not report experiments in which we carefully tuned the weights for SC3. The main function of SC3 in the current study was to find good first-round candidates.

Changing the current weights for $Dice_{LCS}$ and $Dice_{LCCS}$ also changes the order in the recommended characters. We illustrate the results of using three different sets of weights below. We show the alternative formulas for SC3, and the resulting recommended lists for "韻" (yun4) and "捐" (juan1) below. From left to right, recommended characters are listed in the order of descending scores. We do not show the Hanyu pingyin symbols of all of the listed characters to avoid congesting the page with the symbols for Chinese pronunciations.

Original: SC3 = SC2 + 10 × $Dice_{LCCS}$ + 5 × $Dice_{LCS}$.
韻: 損隕圓賞韶嘖遺磧貢員韹蹟賸績積臢瓚潰價漬槓慣債讚鑽賽
捐: 損涓娟扣狷拐揖絹捐搞胡咀硼拮採抬喟撰操捉抿湖喟高亭葫臺豪

Alternative 1: SC3 = SC2 + 10 × $Dice_{LCCS}$ + 0 × $Dice_{LCS}$
韻: 隕損賞圓韶嘖韹蹟賸績遺積磧臢瓚潰價漬槓慣債鑽貢員
捐: 損涓娟扣狷拐揖絹捐搞撰操硼拮採抬喟隕噯韻捉胡抿咀捌湖瑚糊喟

Alternative 2: SC3 = SC2 + 0 × $Dice_{LCCS}$ + 5 × $Dice_{LCS}$
韻: 隕損韶嘖賞圓遺讚韹蹟賸績躓積磧臢瓚潰價漬槓慣債鑽賽貢
捐: 涓娟損狷捐揖拐絹扣拮採抬喟捂唷探授掙啃揮援招搞搖

We can see that there are differences in the lists when we adopted different sets of weights in SC3. However, most, if not all, visually similar characters are included in the lists. Hence, treating these as the first-round candidates, we were satisfied with

the weights that we selected. A more practical mechanism to rank these candidate characters in the context of words will be introduced in Section 4.4. Due to that ranking mechanism, the resulting performances of different weights will not differ significantly for the current application, as long as we have chosen a satisfactory set of weights.

When we focus on just finding just visually similar characters, there will be no contextual information, that is, the words, available to rank the characters. In such cases, the weights certainly matter. Song et al. [2008] discuss related issues when they build a system for Chinese spelling checker. We [Liu et al. 2011] also face a similar problem when we need software to find characters that find Chinese characters that contain specific components.

4. DATA SOURCES AND PRELIMINARY ANALYSES

We provide information about our lexicons, the sources from which we obtained the reported errors in Chinese text, and our analyses of these reported errors in this section.

4.1 Lexicons

For both traditional and simplified Chinese, we prepare a lexicon that provides information on the pronunciation and a database that contains the extended Cangjie codes for the characters. Our programs rely on these databases to generate lists of characters that are phonologically and visually similar to a given character.

It is not difficult to acquire lexicons that contain information about standard pronunciations for Chinese characters. As we stated in Section 2, the main problem is that it is not easy to predict how people in different areas in China and Taiwan actually pronounce the characters. In the current study we employ the standards for Mandarin Chinese that are recorded in the lexicons and published by the official agency in Taiwan[6]. Experimental results reported in Section 5 will show that the ethnic background and mother tones did not influence the performance of our methods very much (at most 1%).

With the procedure reported in Section 3.3, we built databases of extended Cangjie codes for both the traditional and the simplified Chinese. Our database for the traditional Chinese was designed to contain 5,401 common characters in the BIG5 encoding system (between 0xa440 and 0xc67e), which was originally designed for the traditional Chinese. We will call this list of characters TCdict. We converted the traditional Chinese characters to their simplified counterparts and built the database of Cangjie codes for the simplified Chinese. Because two different traditional Chinese characters may be transformed to a common simplified form, this simplified list contains only 5,170 different characters, and we call this list of characters SCdict.

Count from the very first day of the conception of the main ideas, it took us a long time to develop the current TCdict and SCdict. The original idea was published in Liu and Lin [2008], but we continued to try different ideas since then. With the help of the software, that we explained in Section 3.3, to analyze the frequent substrings of the original Cangjie codes, two graduate students (the third and the fourth authors) were able to come up with a good version of the extended Cangjie code for the 5,401 traditional Chinese characters in a couple of weeks. That initial version was modified once in a while afterward. The modification operations were motivated by results of sporadic tests we ran with some data (Elist and Jlist, to be explained in Section 4.2), so

[6]See http://www.cns11643.gov.tw/AIDB/welcome_en.do.

we used some new data (Wlist and Blist, also to be explained in Section 4.2) to examine the performance of our system.

We employed our experience with the traditional Chinese to build the first and only version of the extended Cangjie codes for the simplified Chinese characters in few weeks. Most of the work was conducted only by the second author. We did not run experiments for the simplified Chinese while we are building the extended codes. Therefore, the experimental results that we report in Section 5.6 were not already based on new data.

4.2 Sources of Incorrect Words and Their Roles in Experiments

We acquired five lists of reported errors in Chinese at different stages of our study. By 2009, we collected two lists of errors for traditional Chinese, and in 2010, we added two lists of errors for traditional Chinese and a list of errors for simplified Chinese.

All of these lists contained information about the observed errors. In order to facilitate our experiments, we saved the reported errors in a simple format. An item of a reported error contains three parts: the correct word, the correct character that will be replaced, and the actual incorrect character. For instance, the correct way to write a type of banana is "芭蕉" (ba1 jiao1) and sometimes people use "笆" (ba1) for "芭" (ba1). In this case, we will maintain a data item "芭蕉,芭,笆" for this error.

At the beginning of our study, we acquired two lists of reported errors for traditional Chinese. The first list was obtained from a book published by the Ministry of Education (MOE) in Taiwan [MOE 1996]. The second list was collected in 2008 from the written essays of students of the seventh and the eighth grades in a middle school in Taipei. The errors were entered into computers based on students' writings, not including those characters that did not actually exist and could not be entered. We call the first list of errors the Elist, and the second the Jlist. Elist and Jlist contain, respectively, 1,490 and 1,718 items of errors.

Two or more different ways to write the same words incorrectly were listed in different items and considered as two items. When the same character of a word can be written incorrectly in multiple ways, for example, writing "應付" (ying4 fu4) as "應附" (ying4 fu4) or "應咐" (ying4 fu4) in Jlist, we considered them different errors. Cases like these make a program difficult to find the best actual incorrect character, as we will see in Sections 5.5 and 5.6.

Repeated or semantically related errors were treated as many times as the errors were committed by writers. Writing "變得更好" (bian4 de2 geng4 hao3) as "變的更好" (bian4 de1 geng4 hao3) and writing "變得更強" (bian4 de2 geng4 qiang2) as "變的更強" (bian4 de1 geng4 qiang2) can be considered repeated errors. Writing "變得更好" as "變的更好" and writing "作得不錯" (zuo4 de2 bu2 cuo4) as "作的不錯" (zuo4 de1 bu2 cuo4) can be considered related errors in lexical semantics. (These errors were observed in Jlist.)

These decisions helped us preserve the original distribution of the reported errors. That is, we took the test data as they were and did not try to manipulate or change the reported incorrect Chinese words. However, this also allowed a larger influence of the repeated errors on the reported experiment results.

In order to conduct further experiments, we collected two more lists of errors for traditional Chinese in 2010. The main reason for obtaining these lists was to use them as extra test data for our Cangjie codes that were improved during 2008 and 2009. Since we had access to both Elist and Jlist while we were improving the extended Cangjie codes for TCList, we thought it would be necessary to have new test data that we had never seen before to examine the effectiveness of the improved codes.

Table VI. Quantities of Reported Errors in Different Lists

Data Source	Original	Reduced	Data Source	Original	Reduced
Elist	1490	1333	Wlist	199	188
Jlist	1718	1645	Blist	487	385
			Ilist	684	621

The new datasets were acquired from independent sources. The first new list was collected from the Internet,[7] and the second new list came from errors discussed in a published book that was compiled by scholars [Tsay and Tsay 2003]. The first and the second lists contain 199 and 487 incorrect words, and we refer to these lists as Wlist and Blist, respectively.

In order to test whether our approach works for capturing errors in simplified Chinese, we searched the Internet for reported errors for simplified Chinese, and obtained two lists of errors. The first list[8] came from the entrance examinations for senior high schools in China, and the second list[9] contained errors that were observed at senior high schools in China. We used 160 and 524 errors from the former and the latter lists, respectively. Both of these lists of errors were produced by students at the senior high school levels, so we combined them into one list and refer to the combined list as Ilist.

We dropped some of the reported errors in our experiments because of the current scope of study. Some of the reported errors involved characters that did not belong to TCdict (for traditional Chinese) or SCdict (for simplified Chinese). Since we have extended the Cangjie codes for characters that were included only in TCdict for traditional Chinese and in SCdict for simplified Chinese, we ignored reported errors that did not occur in either TCdict or SCdict. This reduced the sizes of the lists that we collected. Table VI shows the sizes of the original and the reduced lists, respectively, under and the Original and Reduced columns.

4.3 Preliminary Error Analyses

In order to know the main reasons that caused the production of the observed errors, we asked two native speakers to classify the causes of these errors into three categories based on whether the errors were related to phonological similarity, visual similarity, or neither. Since the annotators did not always agree on their classifications, the final results are presented in five categories: P, V, N, D, and B in Table VII. P and V indicate that the annotators agreed on the types of errors to be related to, respectively, phonological and visual similarity. N indicates that both annotators believed that the errors were not due to phonological or visual similarity. D indicates that the annotators believed that the errors were due to phonological or visual similarity, but they did not have a consensus on the category. B indicates the intersection of P and V, that is, errors that are related to both phonological and visual similarities. Table VII shows the percentages of errors in these categories.

We used the quantities of reported errors in the reduced lists as the denominators to compute the percentages in Table VII. Hence, 79.9% in the "Jlist" row indicates that 1,314 (= 1645×0.799) errors were classified as related to phonological similarity. To get 100% for a row in the table, we need to add P, V, N, and D, and subtract B from the total.

[7]See http://www.eyny.com/archiver/tid-2529010.html; last visited on 30 September 2010.
[8]See http://www.0668edu.com/soft/4/12/95/2008/2008091357140.htm; last visited on 10 June 2010.
[9]See http://gaozhong.kt5u.com/soft/2/38018.html; last visited on 30 September 2010.

Table VII. Error Analysis of the Errors: Phonological Influences Dominated in These Errors

	P	V	N	D	B
Elist (traditional)	67.2%	66.1%	0.2%	3.6%	37.1%
Jlist (traditional)	79.9%	30.7%	2.4%	7.9%	20.9%
Wlist (traditional)	69.1%	54.8%	4.8%	8.0%	36.7%
Blist (traditional)	81.6%	34.8%	1.6%	4.7%	22.6%
Ilist (simplified)	83.1%	48.3%	0%	3.7%	35.1%

In all of these five lists, phonological similarity showed a dominant influence in respect to the visual similarity of the reported errors. Most of the reported errors were related to similar pronunciations, while the percentage of errors that were related to visual similarity depended on the lists of the reported errors. It should not be very surprising that the annotators may disagree sometimes.

The weighted proportion of phonologically related errors is 76.0%. Based on the statistics shown in Table VI and Table VII, this analysis considered 4,172 errors (the total of the errors in the reduced lists). The total number of errors that were related to similar pronunciation is $1333 \times 0.672 + 1645 \times 0.799 + 188 \times 0.691 + 385 \times 0.816 + 621 \times 0.831 = 3170.25$. The result of dividing 3170.25 by 4172 is 76.0%. Similarly, we can compute that the weighted proportion of visually related errors is $(1333 \times 0.661 + 1645 \times 0.307 + 188 \times 0.548 + 385 \times 0.348 + 621 \times 0.483) \div 4172 = 46.1\%$.

It is particularly noticeable that although the errors in Jlist were collected from written documents, the phonological factor still dominated. It is a common belief that the dominance of pronunciation-related errors in electronic documents occurs as a result of the common habit of entering Chinese with pronunciation-based methods. The ratio between P and V, that is, P ÷ V, for the Jlist challenges this popular belief and indicates that even though the errors occurred during a writing process, rather than typing on computers, students still produced more pronunciation-related errors. Distribution over error types is not as related to input method as one may have believed. Nevertheless, the observation might still be a result of students in Taiwan being so used to entering Chinese text with a pronunciation-based method that the organization of their mental lexicons is also pronunciation related. The P ÷ V ratio for the Ilist also supports this phenomenon, suggesting that the dominance of phonological influence may be a common phenomenon in the use of both traditional and simplified Chinese. The ratio for the Elist suggests that editors of the MOE book may have chosen the examples with a special viewpoint in their minds—that of balancing pronunciation and composition related errors. (The Blist is so short that we do not consider it representative in regard to this issue.)

It is worthwhile to note that a large percentage of errors are related to either phonological or visual similarity in Chinese. The sum of the statistics under N and D columns indicates the proportion of errors that were related to neither visual nor phonological similarity. The weighted average of (N + D) for the five lists was just 7%. The lowness of this figure can be explained by the large percentage of phono-semantic compounds (xingsheng words, "形聲字") in Chinese.

4.4 Web-Based Statistics

In this section, we examine the effectiveness of using Web-based statistics to differentiate correct and incorrect characters. The abundance of text material on the Internet allows people to treat the Web as a corpus[10]. When we send a query to Google, we will

[10]See http://webascorpus.org.

Table VIII. Reliability of Web-Based Statistics (Based on Data Collected in April 2010)

	Elist			Jlist			Ilist		
	C	A	I	C	A	I	C	A	I
P	92.4%	0.1%	7.5%	91.3%	0.9%	7.8%	97.1%	0.0%	2.9%
V	92.6%	0.0%	7.4%	91.5%	0.6%	7.9%	98.0%	0.0%	2.0%

be informed of the estimated number of pages[11] (ENOPs) that possibly contain relevant information. If we put the query terms in quotation marks, we should find the Web pages that replicate the query forms in the exact sequence and with the same adjacency as those originally entered. Hence, it is possible for us to compare the ENOPs for two competing phrases for guessing the correct way of writing a word. For instance, at the time of this writing, Google reported 116,000 and 33,000 relevant pages, respectively, for "strong tea" and "powerful tea". (When conducting such advanced searches with Google, the quotation marks are needed to ensure the adjacency of the individual words.) Hence, "strong" appears to be a better choice to go with "tea". This is an idea similar to one of the approaches for computing collocations based on word frequencies [cf., Manning and Schütze 1999]. Although the idea may not work very well when using a small database, the size of the current Web should be large enough.

We ran experiments for only those items that the annotators were in consensus over the causes of the error. Hence, for instance, we had 1285(= 1333 × (1-0.036), cf. Table VI and Table VII), 1515 (= 1645 × (1-0.079)), and 598(= 621 × (1-0.037)) such words for Elist, Jlist, and Ilist, respectively. As the information available on the Web may change over time, we also have to note that the statistics reported in Table VIII were based on experiments conducted during April 2010.

Table VIII shows the results of our investigation. For each reported error, we submitted the correct word and the incorrect word to Google and considered that we had a correct result when we found that the ENOP for the correct word was larger than the ENOP for the incorrect word. If the ENOPs were equal, we recorded an ambiguous result; and when the ENOP for the incorrect word was larger, we recorded an incorrect event. We use C, A, and I to denote correct, ambiguous, and incorrect events, respectively, in the table. We record a correct result for the "strong tea vs. powerful tea" test, for instance.

The Web-based statistics did not work very well for the Elist and Jlist, but seemed to work well enough for Ilist. The most common reason for the errors is that certain words are confusing to the extent that the majority of the Web pages showed the incorrect words. Some of the errors are so common that even one of the Chinese input methods on Windows XP offered wrong words as possible choices, for example, "雄赳赳" (xiong2 jiu1 jiu1; the correct one) vs. "雄糾糾" (xiong2 jiu1 jiu1). It is also interesting to note that people may intentionally use incorrect words on some occasions; for instance, people may choose to write homophones in advertisements.

Another possible reason for the mistakes is that whether a word is correct depends on a larger context. For instance, "小斯" (xiao3 si1) is more popular than "小廝" (xiao3 si1) because the former is a popular nickname. Unless we provided more contextual information about the queried words, checking only the ENOPs of "小斯" and "小廝" would lead us to choose "小廝", which would be an incorrect word if we meant to find the right way to write "小斯". Other difficult pairs of words to distinguish are"紀錄" (ji4 lu4) vs. "記錄" (ji4 lu4) and "須要" (xu1 yao4) vs. "需要" (xu1 yao4).

[11]According to Croft et al. [2010], the ENOPs may not reflect the actual number of pages on the Internet, they may result from statistical estimations.

Yet another reason for having a large ENOP for the incorrect words was due to errors in segmenting Chinese character strings (cf., Ma and Chen [2003]). Consider a correct character string "WXYZ". It is possible that "XY" happens to be an incorrect way to write a correct word. This is the case for having the counts for "花海繽紛" (hua1 hai3 bin1 fen1) to contribute to the count for "海繽", which is an incorrect form of "海濱" (hai3 bin1).

A reason why the Web-based statistics worked for Ilist is that all of the correct words in Ilist contained four characters. None of the factors that we list above for explaining the errors that we observed from Elist and Ilist may reasonably apply in the case of four-character strings. Hence, Web-based statistics worked almost perfectly for Ilist.

We compared the statistics reported in Table VIII and our reports in Liu et al. [2009c] and found that the effectiveness for Elist and Jlist improved greatly. To understand this phenomenon, we examined the records of our experimental results. Let x and y be the ENOPs of the correct and incorrect words, respectively. When we reported cases in which our systems failed to identify the correct character, that is, the I cases, in 2009, many of the ratios of y against x were within approximately 5%. Namely, we had $(y/x) \leq 1.05$in many cases, indicating that the margins were very small in 2009. The differences between the ENOPs may have changed between 2009 and 2010, so making us to achieve the better performance reported in Table VIII.

The improvement shows that the reliability of the Web-based statistics may be improving as the correct usage of words increase. This also shows that our approach might not work very well if the majority of Web-page authors do not use the characters in standard ways. To avoid the problem of using only the raw values of ENOPs to judge the correctness of words, we will introduce an alternative method in the next section.

5. EXPERIMENTAL EVALUATION

We evaluate the effectiveness of using the phonologically and visually similar characters to capture errors in Chinese words in this section.

In Section 5.1, we provide details about the procedures for the experiments, and, in Section 5.2, we explain the definitions of inclusion rates that we used to evaluate the basic performance of our systems. In Section 5.3, we offer a comparison of the statistics about our experiments that were conducted in 2009 and 2010, in order to gather information about the reliability of Web-based statistics, which is crucial for the success and applicability of our systems in the long run. In Section 5.4, we report the inclusion rates of our systems on the five sources of test data. The experiments also show the robustness of our methods and data for processing test data that were not reported in previous conference articles. In Sections 5.5 through 5.7, we deepen and widen our investigation of the effectiveness of our systems by applying an alternative method to rank the candidate characters and by recommending a limited number of candidates characters based on two ranking mechanisms.

5.1 Experimental Procedure

We designed and employed the ICCEval procedure for the evaluation task. We needed two types of data for the experiments. The information about the pronunciation and structures of the Chinese characters (Section 4.1) helped us generate lists of similar characters. We also needed reported errors (Section 4.2) so that we could evaluate whether the similar characters catch the reported errors.

At step 1, we created a list of characters based on the selection criterion, given the correct word and the correct character to be replaced. We may choose to evaluate the

Procedure ICCEval
Input:
A test item that includes the following information:
(Section 4.2)
cwd: the correct word,
ccr: the correct character,
aic: the actual incorrect character;
crit: the selection criterion;
num: number of requested characters;
rnk: the criterion to rank the incorrect words;
Output: a list of ranked candidates for **ccr**
Steps:
1. Select a candidate list, L, of characters, from TCdict or SCdict, for **ccr** with the specified criterion, **crit**. When using SC1, SC2, or SC3 to select visually similar characters, at most **num** characters will be selected.
2. Check whether **aic** belongs to L (the inclusion test). if yes, continue; else, return nil.
3. For each x in L, replace **ccr** in **cwd** with x, submit the resulting x and incorrect word to Google, and record the ENOPs for x and the incorrect word.
4. Rank the characters in L with the criterion specified by **rnk**.
5. Return the ranked list.

effectiveness of phonologically or visually similar characters. For a given correct character, ICCEval can generate characters that are in the SS, SD, MS, and MD categories for phonologically similar characters (Section 2). For visually similar characters, ICCEval can select characters based on different score functions, i.e., SC1, SC2, and SC3 (Section 3.4). In addition, ICCEval can generate a list of characters that belong to the same radical and have the same number of strokes with the correct character. In the experimental results, we refer to this type of similar characters as RS.

At step 2, we checked whether the selected list of characters indeed contained the actual incorrect character.

At step 3, for a correct word that people should write, we replaced the correct character with a character from the candidate list that was generated at step 1, submitted the incorrect word to Google AJAX Search API (or directly to the Google interface[12], and extracted the ENOP of pages that contained the incorrect words. In an ordinary interaction with Google, an ENOP can be retrieved from the search results, and it typically follows the string "Results 1-10 of about" in the browser window. Using the Google AJAX Search API, we have only to parse the returned results using a simple method.

Larger ENOPs for incorrect words suggest that these words are incorrect words that people frequently used on their Web pages. Hence, we could rank the similar characters based on their ENOPs at step 4 and return the list.

Since the reported errors contained information about the actual incorrect ways to write the correct words (Section 4.2), we could check whether the actual incorrect characters were among the similar characters that our programs generated at step 2 (inclusion tests). We could also check whether the actual incorrect characters were ranked higher in the ranked lists (ranking tests).

Take the word "和藹可亲" (he2 ai3 ke3 qin1) as an example. In the collected data, it was reported that people wrote this word as "和靄可亲" (he2 ai3 ke3 qin1), that is, the second character was incorrect. Hence the test item (correct word, correct character, actual incorrect character) is ("和藹可亲", "藹", "靄"). Hoping to capture the error, ICCEval generated a list of possible substitutions for "藹" at step 1. Depending on the

[12]See http://www.google.com.

categories of sources of errors, ICCEval generated a list of characters. When aiming to test the effectiveness of visually similar characters, we could ask ICCEval to apply SC3 to selected a list of alternatives for "遏" from SCdict, and the results may include "靄" (ai3), "谒" (ye4), "葛" (ge3), and other candidates. At step 2, we found that the actual incorrect character was included in the candidate list. At step 3, we created and submitted the query strings "和靄可亲", "和谒可亲", and "和葛可亲" to obtain the ENOPs for the candidates. If the ENOPs were, respectively, 571,000, 445,000, and 8,580, these candidates would be returned in the order of "靄", "谒", and "葛". As a result, the returned list contained the actual incorrect character "靄", and placed "靄" at the top of the ranked list.

Notice that we considered the contexts in which the incorrect characters appeared to rank the candidate characters. We did not rank the incorrect characters with the unigrams such as "靄", "谒", and "葛" alone. Instead, the candidates were ranked with the ENOPs of "和靄可亲", "和谒可亲", and "和葛可亲".

In addition, although this running example shows that we ranked the characters directly with the ENOPs, we also tried to rank the list of alternatives with the pointwise mutual information [PMI; cf., Jurafsky and Martin 2009]:

$$PMI\,(C, X) = \frac{\Pr(C \wedge X)}{\Pr(C) \times \Pr(X)}, \tag{2}$$

where X is the candidate character to replace the correct character and C is the correct word excluding the correct character to be replaced. To compute the score for the replacement of "遏" with "靄" in "和遏可亲", X = "靄", C="和□可亲", and $(C \wedge X)$ is "和靄可亲". (□ denotes a character to be replaced.) We chose to try the frequency-based method and PMI-based method because both are used to compute the strength of collocation in natural language processing [Manning and Schütze 1999].

It would demand a considerable amount of computation effort to find $\Pr(C)$ in general, if this is a required task. Fortunately, we do not have to consider the effect of $\Pr(C)$ because it is a common denominator for all possible incorrect characters for a given incorrect word. Let X_1 and X_2 be two competing incorrect characters for the correct character. We can ignore $\Pr(C)$ because of the following relationship.

$$PMI\,(C, X_1) \geq PMI\,(C, X_2) \Leftrightarrow \frac{\Pr(C \wedge X_1)}{\Pr(X_1)} \geq \frac{\Pr(C \wedge X_2)}{\Pr(X_2)} \tag{3}$$

Hence, X_1 prevails if $score\,(C, X_1)$ is larger, where $score\,(C, X)$ for any$X$is listed in Equation (4).

$$score\,(C, X) = \frac{\Pr(C \wedge X)}{\Pr(X)} \tag{4}$$

In our work, we approximate the probabilities used in Equation (4) by the corresponding frequencies. Namely, we replace $\Pr(C \wedge X)$ with the Web-based counts for $(C \wedge X)$, for example, "和靄可亲"; and substitute $\Pr(X)$ with the Web-based counts for X, for example, "靄". The counts were obtained with exactly the same mechanism that we used to obtain the ENOPs that we explained at the beginning of this section.

5.2 Performance Measures

Recall that we used only those errors for which the annotators had reached consensus on the causes of the errors. The errors that involved characters that were not in TCdict and not SCdict were not considered in the current study either.

Given the errors in the lists in Table VI, we ran ICCEval**,** and measured the performance in two ways. First, we would like to have the candidate list (step 1 in ICCEval)

include the actual incorrect character in the inclusion test. Second, we would prefer that the actual incorrect character be placed at the top of the ranked list (step 4 in ICCEval).

Assume that there were n items of errors in a given list and that the candidate lists for these n errors contained m of the n actual incorrect character. Then, we compute the inclusion rates in the following manner.

$$\text{inclusion rate} = \frac{m}{n} \tag{5}$$

In order to compare whether it is easier to capture either the phonologically similar or the visually similar errors, we separate the test instances according to the annotators' consensuses. Hence, we will provide separate inclusion rates for two types of error. When reporting the inclusion rates for phonologically similar errors, we use the number of phonologically-similar errors in place of n, and when reporting the inclusion rates for visually-similar errors, we use the number of visually-similar errors in place of n.

The inclusion rates are similar to the recall rates that are commonly-used in the field of information retrieval (e.g., Croft et al. [2010]). Yet, they are different. The recall rates measure how well a search engine retrieves the correct answers for a query. In our experiments, each test has only one answer, that is, the actual incorrect character. Hence, we measure the probability whereby we would capture the actual incorrect character across the reported errors.

In addition, we prefer to put the actual incorrect character higher in the ranked list. Hence, we analyzed the accumulative inclusion rates of the top k characters in the candidate list. Assume that there were n items of errors in a given list. Also assume that ICCEval placed the actual incorrect characters at the j^{th} position t_j times in the n experiments, where the first position is the best position. The ratio ($t_j \div$n) is the probability that the actual incorrect character was ranked at the j^{th} position in an individual test. Then, we compute the accumulative inclusion rate (AIR) in the following way.

$$\mathrm{R}_k = \frac{\sum_{j=1}^{k} t_j}{n} \tag{6}$$

If ICCEval returns the candidate list completely, then it will achieve the inclusion rate. If ICCEval returns only the top k candidates, it will achieve the accumulative inclusion rates, which are upper-bounded by the inclusion rate.

5.3 Temporal Comparison

In April 2010, we repeated the experiments that we conducted in March 2009. The main purpose was to inspect whether we would achieve performance of similar quality with Web-based statistics in 2009 and 2010. The experiments had two goals: (1) to have the candidate lists (step 1 in ICCEval) include the incorrect characters in the reported errors, and (2) to put the actual incorrect characters at top of the ranked results (step 4). We used Elist and Jlist in the experiments.

Because we kept the lists of candidate characters that we generated at step 1 in ICCEval in 2009, we were able to resubmit the incorrect words at step 3 and collected the ENOPs to rank the incorrect characters in 2010. In 2009, we submitted our queries directly through the Google interface[13] and we repeated the same procedure in the experiments in 2010 again. (Submitting a large amount of queries directly to the standard Google interface resulted in Google considering our programs to be a malicious

[13]See http://www.google.com.

Table IX. Inclusion Rates in 2009 [Liu et al. 2009b]

	SS	SD	MS	MD	SC1	SC2	RS	Phone	Visual	All
Elist	91.6	18.4	3.0	1.9	76.1	73.9	4.1	99.0	82.0	93.4
Jlist	92.2	20.2	4.2	3.7	74.5	67.5	6.1	99.3	77.6	97.3

attacking agent, and our computers might be blocked. Hence, we have switched to the Google AJAX search API for the other new experiments.)

Since we reused the candidate lists, the inclusion rates did not change over time, so we show the inclusion rates in Table IX. We show the inclusion rates of the candidate lists that were generated with different criteria, that is, SS or SC1 for Elist and Jlist. We did not run SC3 for this experiment because we did not have this score function in 2009. Using SS to recommend candidate characters, we captured 91.6% of the errors that were related to pronunciation in Elist. Since in Table VII, 67.2% of the errors in Elist was related to phonological similarity, using the SS list alone captured (91.6 $\times$ 67.2)%, that is, 61.6%, of all of the errors. Using SC1 to recommend candidate characters, we captured 74.5% of the errors that were caused by visual similarity, and that is (74.5 $\times$ 66.1)%, that is, 49.2%, of all of the errors in Elist. The Phone column shows the inclusion rates when we used the union of the recommend lists created with SS, SD, MS, and MD criteria to guess the actual incorrect characters that were caused by phonological similarity. The Visual column shows the inclusion rates when we used the union of the candidate lists created with SC1, SC2, and RS criteria to guess the actual incorrect characters that were caused by visual similarity. The All column shows the inclusion rates when we used the union of the candidate lists created with SS, SD, MS, MD, SC1, SC2, and RS criteria to guess all of the actual incorrect characters.

Apparently, it was much easier to capture errors that were related to the phonological similarity. All together, we were able to capture more than 95.5% of the nearly 2,978 observed errors. (cf., Table VI and Table IX; 95.5 = (93.4 $\times$ 1333 + 97.3 $\times$ 1645) $\div$ (1333 + 1645)) Recall that SS stands for "same sound and same tone." SS is the most effective criterion to use to select a candidate list that can capture the observed errors. Even so, candidate lists selected with other criteria were necessary to achieve 99% inclusion rates for the phonologically related errors.

In contrast, using the extended Cangjie code that we constructed manually in 2009 did not perform comparatively well for visually-related errors, although achieving an inclusion rate of 80.4% in 2009 was very encouraging. It is worth mentioning that the RS category was able to capture 4.8% of the visually similar errors. (cf., Table VI, Table VII, and Table IX; 80.4 = (82.0 $\times$ 1333 $\times$ 0.661 + 77.6 $\times$ 1645 $\times$ 0.307) $\div$ (1333 $\times$ 0.661 + 1645 $\times$ 0.307). We used only those errors that were caused by visual similarity in this calculation.)

Table X shows the accumulative inclusion rates up to the 10^{th}-ranked candidates for the errors in Elist. If we subtract the results for 2009 from the corresponding numbers for 2010 in the table, we will see that there is no significant difference between the data shown in the upper and the lower part of Table X. We also reran the tests for Jlist, and did not observe any significant differences in statistics for the ranked lists in 2009 and 2010, either. Web-based statistics therefore appeared to be very stable, for the task of ranking the candidate incorrect characters.

5.4 Inclusion Tests

We have changed several aspects of our system since we conducted the experiments reported in the previous subsection. We have built a new version of the extended Cangjie codes for the characters in TCdict and the first version of the extended Cangjie codes for the characters in the SCdict, using the procedure that we discussed in Section 3.3.

Table X. AIRs for Ranking the Candidates for Elist Based on the Frequencies Collected in 2009 and 2010

		R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
March 2009	SS	55.5	74.6	82.4	85.8	87.6	89.3	90.1	90.7	90.7	91.1
	SD	10.0	13.8	15.4	16.3	16.6	17.1	17.4	17.6	17.9	18.3
	MS	2.1	2.3	2.5	2.7	2.9	3.0	3.0	3.0	3.0	3.0
	MD	1.1	1.4	1.4	1.4	1.4	1.4	1.7	1.7	1.7	1.7
	SC1	33.6	49.4	55.9	61.4	65.8	67.7	68.8	71.2	72.0	73.2
	SC2	30.6	43.5	50.7	56.8	60.6	64.1	65.5	67.1	68.7	70.1
	RS	2.7	3.5	3.7	4.0	4.1	4.1	4.1	4.1	4.1	4.1
April 2010	SS	55.1	73.4	81.7	86.0	88.4	90.0	90.4	90.9	91.1	91.2
	SD	10.0	13.2	15.1	16.1	16.6	16.9	17.3	17.6	17.8	17.9
	MS	2.1	2.5	2.8	2.8	2.9	2.9	3.0	3.0	3.0	3.0
	MD	1.0	1.4	1.4	1.4	1.6	1.6	1.6	1.6	1.7	1.7
	SC1	31.1	48.0	57.1	62.8	66.6	68.4	69.9	70.8	72.7	73.7
	SC2	29.4	43.7	52.1	58.5	62.7	64.5	65.9	67.0	68.6	69.6
	RS	2.8	3.5	3.8	3.8	4.0	4.0	4.0	4.1	4.1	4.1

We have also added a new score function, that is, SC3, that we did not have in 2009. Furthermore, we do not submit our queries to the ordinary Google interface anymore, as we explained in Section 5.3.

To make the results of the experiments with traditional Chinese more convincing, we used two new lists, Wlist and Blist that we did not know when we improved the Cangjie codes in TCdict. These two lists serve as unforeseen test instances for our programs and databases.

We ran ICCEval with Elist, Jlist, Wlist, Blist, and Ilist. The experiments were conducted for all categories of phonological and visual similarity. When using SS, SD, MS, MD, and RS as the selection criteria, we did not limit the number of candidate characters. Any characters that conform to the selection criteria were selected in the candidate list, L in ICCEval. When using SC1, SC2, and SC3 as the selection criteria, we limited the number of candidates to be no more than 30. We inspected samples of the candidate lists generated with SC1, SC2, and SC3, and found that the number of visually similar characters for a given character rarely exceeded 30. Hence this limit was chosen heuristically.

We considered only words that the native speakers were in consensus over the causes of errors. There is a limit on the maximum number of queries that one can submit to the Google AJAX API. As a consequence, we could not complete our experiments in a short time, and the ENOPs were obtained during March and April 2010. Table XI shows the inclusion rates that the candidate lists, generated with different crit at step 1, contained the incorrect characters in the reported errors. The columns have the same meaning as they have in Table IX. The rows show the statistics that we observed while using the lists, listed in Table VI, as the test data. For instance, we achieved an inclusion rate of 90.3% for the visually similar errors when we applied SC3 to generate the candidate lists for errors in the Wlist.

`ICCEval` and our databases worked well for traditional Chinese. Although we have slightly expanded the definitions of similar sound since 2009, the effectiveness of SS, SD, MS, and MD remain the same for Elist and Jlist in Table IX and Table XI. The statistics for RS did not change because we were using the same list in 2009 and in 2010. Statistics about SC1 and SC2 are slightly better in Table XI for both Elist and Jlist, but the improvements are not significant. Using the new score function, that is,

Table XI. Inclusion Rates for the Different Experiments

	SS	SD	MS	MD	SC1	SC2	SC3	RS	Phone	Visual	All
Elist	91.6	18.4	3.0	1.9	77.7	76.3	87.3	4.1	99.0	89.8	96.2
Jlist	92.2	20.2	4.2	3.7	77.0	71.3	89.3	6.1	99.3	91.9	99.3
Wlist	94.6	23.1	0.8	0.8	80.6	78.6	90.3	1.1	99.2	90.3	96.9
Blist	82.2	24.2	3.2	1.9	77.6	75.4	90.3	3.7	95.2	91.8	94.7
Ilist	82.6	29.3	2.1	1.6	78.3	71.0	87.7	1.3	97.3	90.0	96.5

SC3, and the new Cangjie codes significantly improved the inclusion rates for visually related errors. We were able to include 88.3%[14] of the visually similar errors with SC3. We were able to include approximately 10% more of the actual incorrect characters in experiments, when we used SC3 rather than SC1 or SC2 to generate the candidate lists.

Even though we had not seen the errors in Wlist and Blist[15] previously, our program had shown a robust performance. ICCEval achieved a comparable performance when running with Wlist than running with Elist and Jlist. When working with Blist, ICCEval did not perform as well with phonologically similar errors, but showed a similar performance for visually similar errors. However, the change was not very significant, and the results reflected the preference of the experts who wrote the book from which we obtained Blist. The effectiveness of the SS lists dropped, while the effectiveness of the SD lists increased. We inspected the errors in Blist closely, and found many challenging instances—those that native speakers found the incorrect words are becoming more frequently used in practice. For this reason, we considered that ICCEval achieved reasonably well.

When running with Ilist, ICCEval achieved a performance similar to the one which it had achieved with Blist. Like the results observed in the other experiments, it is easier to find phonologically similar incorrect characters than visually similar ones. Using SS and SC3 as the selection criterion at step 1 in ICCEval were the most effective criteria for phonologically and visually similar characters, respectively. The contribution of SD was quite significant for Ilist.

When we used the unions of the phonologically similar characters to compute the inclusion rates, we captured 98.6% of the phonologically similar errors for the five lists. The unions of the visually similar characters were also very effective, though capturing only about 90.5% of the visually similar errors. When we used the union of all of the candidate lists, we captured 97.4% of all the errors. These are the weighted averages of the inclusion rates which we calculated with a procedure similar to the one provided in the tenth footnote.

It is certainly desirable for applications to have the potential to capture all of the reported errors. However the inclusion rates were achieved at different costs. For each reported error and the actual incorrect character of the error, ICCEval generated a candidate list at step 1. In an experiment that used a list of y errors, we would have y candidate lists. Hence, for a particular experiment, we can calculate the average length of such $y$candidate lists. Table XII shows the average lengths of the corresponding experiments that are reported in Table XI.

[14]The rates are averages computed considering the numbers of errors in the error lists, listed in Table VI and Table VII. For instance, 88.3 = (87.3 × 1333 × 0.661 + 89.3 × 1645 × 0.307 + 90.3× 188 × 0.548 + 90.3 × 385 × 0.348 + 87.7 × 621 × 0.483) ÷ (1333 × 0.661 + 1645 × 0.307 + 188 × 0.548 + 385 × 0.348 + 621 × 0.483).

[15]We used only 20 of the reported errors in Blist in Liu et al. [2009a].

Table XII. Average Lengths of the Candidate Lists

	SS	SD	MS	MD	SC1	SC2	SC3	RS	Phone	Visual	All
Elist	11.3	18.6	9.7	21.8	23.3	26.7	25.4	9.2	56.6	48.8	102.1
Jlist	12.4	22.0	11.6	25.4	21.9	24.3	25.4	7.7	64.3	46.0	107.4
Wlist	11.8	17.4	10.7	22.0	22.6	26.1	25.5	9.2	56.4	48.8	102.0
Blist	14.2	22.2	10.4	22.5	22.2	25.6	25.7	8.1	62.5	47.4	106.9
Ilist	12.6	19.1	9.1	19.5	24.3	27.1	25.5	9.4	55.5	47.8	100.2

Clearly, longer candidate lists would increase the chances to achieve higher inclusion rates. Hence, it would be more preferable if a shorter candidate list can achieve the same inclusion rate as that of a longer candidate list. From this perspective, the statistics in Table XII show that SS is very effective for capturing phonologically-similar errors, as we were able to capture better than 89.8% of the phonologically-similar errors by an average of 12.2 characters. (12.2 is the weighted average of 11.3, 12.4, 11.8, 14.2, and 12.6.) Taking the union of SS, SD, MS, and MD lists to obtain the "Phone" list, the average lengths increased from 12.2 to 60.0, but the inclusion rates increased from 89.8% only to 98.5%.

Using SC3 as the selection criterion for visually similar errors offered a significant improvement in both effectiveness and efficiency. The weighted average lengths of the candidate lists that were selected with SC1, SC2, and SC3 were 22.8, 25.7, and 25.4, respectively. The weighted average inclusion rates for SC1, SC2, and SC3 were 77.6%, 73.6%, and 88.6%, respectively. Using SC3 allows us to achieve higher inclusion rates with shorter candidate lists. We took the union of the SC1, SC2, SC3, and RS lists to form the Visual list, increased the average lengths of the candidate lists to 47.4, but increased the inclusion rates only marginally to 90.9%.

To achieve the inclusion rates in the All column in Table XI, we would have to allow ICCEval to recommend 104.3 characters. Although a list of 104 characters will be too long to be practically useful, we have to keep in mind that we had reduced the search space from more than 5,100 characters to approximately 100 characters—which is 98% for the compression rate. This point is particularly important to bear in mind as we seek to applying our findings to help teachers select "attractive incorrect characters" when authoring test items of the ICC tests.

5.5 Ranking Tests with Elist and Jlist

To make the candidate lists applicable, we wish to place the actual incorrect character high in the ranked list. This will help the efficiency in supporting computer-assisted test-item writing. Having shorter lists that contain relatively more confusing characters may facilitate the data preparation for psycholinguistic studies as well.

Table XIII shows the results when we recommended only the leading ten candidates for the errors in Jlist. The table is divided into two parts. The upper part (with row heading "Frequency") shows the results when we used the raw values of the ENOPs to rank the candidate characters, and the lower part (with row heading "PMI") shows the results when we used Equation (4) in Section 5.1 to rank the candidate characters. The column "R_i" shows the accumulative inclusion rates (AIRs) that we defined in Equation (6) in Section 5.2. The sub-row headings show the selection criteria that were used in the experiments. For instance, using SS as the criterion and ranking with the raw values of ENOPs, 55.1% of phonologically related errors were included if we offered only one candidate, 70.6% of the phone-related errors were included if we offered two candidates, etc. If we recommended only the top five candidates in SS (ranked with ENOPs), we captured the phonologically similar errors 84.3% of the time. For errors that were related to visual similarity, recommending the top five candidates

Table XIII. AIRs for Ranking the Candidates for Jlist Based on ENOPs and PMIs

		R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{num}
Frequency	SS	55.1	70.6	77.6	81.7	84.3	86.2	87.6	88.2	89.2	89.7	92.2
	SD	10.6	13.8	15.9	16.9	17.7	18.0	18.1	18.5	18.7	18.8	20.2
	MS	3.0	3.3	3.5	3.7	3.7	3.7	3.7	3.8	3.8	3.9	4.2
	MD	2.2	2.7	2.9	3.0	3.1	3.3	3.3	3.3	3.3	3.3	3.7
	Phone	42.9	57.5	64.6	70.1	73.7	77.7	81.1	83.0	84.6	86.1	99.3
	SS+SD	43.7	56.3	64.9	71.7	75.5	78.7	81.1	83.2	84.4	85.5	95.1
	SC1	40.2	53.5	59.8	64.2	65.9	67.7	68.3	69.1	70.5	72.1	77.0
	SC2	34.7	48.5	52.1	55.4	57.6	60.0	62.0	63.0	64.4	65.5	71.3
	SC3	42.6	56.6	64.4	69.7	73.9	77.0	78.8	80.4	81.6	83.6	89.3
	RS	5.3	5.9	5.9	5.9	5.9	5.9	5.9	5.9	6.1	6.1	6.1
	Visual	35.3	50.4	57.9	61.5	66.1	69.2	72.4	74.0	76.0	77.6	91.9
PMI	SS	47.0	62.2	70.7	75.8	79.0	82.8	84.1	85.5	86.9	87.9	92.2
	SD	9.4	12.0	14.1	15.3	15.8	16.6	16.8	17.7	18.2	18.4	20.2
	MS	2.7	3.3	3.3	3.6	3.6	3.6	3.7	3.7	3.7	3.7	4.2
	MD	2.1	2.5	2.7	2.8	2.8	3.0	3.0	3.0	3.1	3.1	3.7
	Phone	37.0	51.0	59.7	64.5	69.0	72.7	75.6	77.7	79.3	80.5	99.3
	SC1	38.2	49.9	56.4	59.4	63.0	66.1	67.5	69.1	70.5	71.7	77.0
	SC2	33.7	45.0	50.1	54.9	57.0	59.0	61.8	63.4	64.4	64.6	71.3
	SC3	39.6	54.3	63.2	69.5	73.9	75.4	78.2	79.4	80.6	82.4	89.3
	RS	3.8	5.9	5.9	5.9	5.9	5.9	5.9	5.9	6.1	6.1	6.1
	Visual	34.7	47.4	56.9	61.9	66.9	69.8	72.8	76.4	77.0	78.4	91.9

in SC3 (ranked with PMIs) would capture the actual incorrect characters 73.9% of the time. As we explained in Section 5.2, the AIRs must be smaller or equal to the inclusion rates of the individual experiments. In Table XIII, we copy the inclusion rates of the Jlist row in Table XI into the R_{num} column.

The statistics listed in Table XIII show the effectiveness of our ranking mechanisms – both ENOPs and also PMIs. The difference (R_{num}-R_i) is a good indicator of the degree of sacrifice required in the situation when we recommend only the top i candidate rather than the complete candidate lists. When we shortened the candidate lists to contain less than 10 characters, we did not sacrifice the inclusion rates significantly. When we recommended 10 characters, the differences (R_{num}-R_{10}) were not large, especially when we considered that we would have to put forward much longer lists of candidate characters, for example, Table XII, to achieve R_{num}. One exception to this observation is that providing the complete candidate lists that were selected with the SS criterion may be worthwhile. According to Table XII, suggesting an average of 12.4 characters achieved R_{num}.

Recall that using the union of the candidate lists, such as Phone and Visual in Table XII, helped us to achieve higher inclusion rates. Although higher inclusion rates are desirable, the detailed statistics in the Phone and Visual sub-rows in Table XIII shed light on the drawbacks of the union lists. If we present only the top k candidates to those who need the similar characters of a given character, the union of the lists might not provide better performance profiles than that of the individual lists, separately.

It is not very difficult to understand the potentially inferior performances of the union lists. Assume that the rank of the actual incorrect character is j in a list, say LL. This implies that there already are at least $(j-1)$ characters that are mistakenly considered as better candidates by the score functions. After we put the lists together

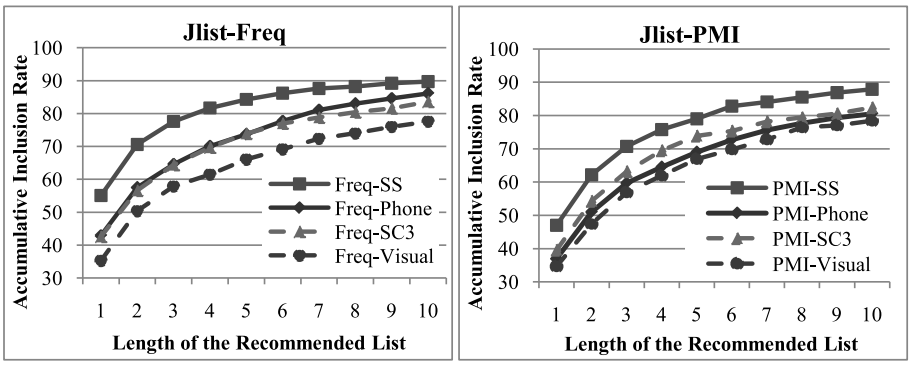


Fig. 3. AIRs of the union lists might not be as good as individual lists.

and rank the joined lists, these $(j-1)$ characters still win against the actual incorrect character. In addition, other characters that were not in LL might be ranked higher than the actual incorrect character in the union. As a consequence, the rank of the actual incorrect character might not improve in the union lists.

Consider, in one particular test, two ranked lists $SS = \{A, B\}$ and $SD = \{C, D\}$, where A, B, C, and D are four different characters. Hence, we must have score $(A) \geq$ score(B) and score$(C) \geq$ score (D). Assume that B is the actual incorrect character, that score $(C) \geq$ score(B), and that score$(B) \geq$ score(D). The union of SS and SD will be $\{A, C, B, D\}$. The rank of the actual incorrect character will drop from 2 in SS to 3 in the union. This is a situation in which the joined list might not outperform the best individual list.

However, it remains possible that the joined lists perform better than the individual ones. This could happen when the actual incorrect characters were included in only one of the individual lists and when the ranks of the actual incorrect characters remain the same in the joined lists. If, in the previous example, score(B) is larger than score(C), than the joined lists will perform as well as SS. Moreover, if, in another test, we have $SS = \{E, F\}$, $SD = \{G, H\}$, where E, F, G, and H are different characters, and where G is the actual incorrect character, than the joined list will perform better than SS.

Given the reasons provided in the previous two paragraphs, we cannot tell whether or not the joined lists will perform better than the best performing individual lists.

Figure 3 shows four pairs of examples for the experiments with Jlist. It happened that the joined lists did not perform as well as the best-performing individual lists in the joined lists. The charts were drawn based on the statistics listed in Table XIII. For instance, the curve "Freq-SS" in the chart with the title "Jlist-Freq" was based on the data in the sub-row "SS" in the "Frequency" part in Table XIII. The performance profiles of SS lists dominated those of Phone lists, and the performance profiles of SC3 lists dominated those of Visual lists in these cases. When we ranked the candidate characters with PMIs, the results were similar, and are shown in the chart titled "Jlist-PMI."

It is interesting to explore whether we may improve the performance of the Phone list by not considering the characters that were in the MS and MD lists. We conducted such an experiment, and in the middle of Table XIII, the row SS + SD shows the AIRs of the union list of words that were formed by using the candidate characters originally in the SS and SD lists. We can compare the performances of the Phone list and the SS + SD list. Overall, the SS + SD list provides better, but not significantly better, performance.

Table XIV. AIRs for Ranking the Candidates for Elist Based on ENOPs and PMIs

		R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
Frequency	SS	55.2	72.7	81.9	85.2	87.5	88.9	89.7	90.0	90.5	90.7
	SD	11.3	14.4	15.6	16.2	16.8	17.2	17.6	17.7	17.8	17.9
	MS	2.0	2.2	2.6	2.7	2.9	2.9	2.9	2.9	3.0	3.0
	MD	1.3	1.6	1.6	1.6	1.6	1.6	1.8	1.8	1.8	1.8
	Phone	39.7	57.1	68.0	72.8	77.1	79.9	82.5	84.6	86.0	87.1
	SS+SD	46.7	62.8	73.5	78.9	83.7	86.2	88.6	89.9	90.6	92.0
	SC1	32.4	46.8	55.7	62.0	65.1	67.7	69.4	71.0	72.7	73.1
	SC2	27.7	41.6	49.2	55.3	59.2	62.6	64.9	66.8	67.9	69.5
	SC3	33.6	49.3	57.6	63.2	68.4	71.4	74.4	77.7	79.2	81.5
	RS	2.8	3.6	3.6	3.7	3.8	4.0	4.0	4.1	4.1	4.1
	Visual	27.7	41.7	49.1	55.1	59.8	63.7	66.7	69.3	71.7	74.1
PMI	SS	51.8	73.9	82.2	85.6	88.1	89.0	89.6	89.7	90.2	90.5
	SD	10.5	14.5	16.1	16.9	17.3	17.3	17.4	17.6	17.8	17.8
	MS	1.7	2.1	2.6	2.6	2.6	2.7	2.8	2.8	2.9	2.9
	MD	1.1	1.6	1.7	1.7	1.7	1.8	1.8	1.8	1.9	1.9
	Phone	40.5	61.6	72.5	77.7	81.6	84.5	86.6	88.2	89.6	91.0
	SC1	35.5	50.5	59.1	63.7	67.7	69.8	71.0	72.2	73.1	74.4
	SC2	32.5	47.1	53.8	58.6	62.5	66.1	67.9	69.6	70.4	71.3
	SC3	36.4	53.2	64.1	69.4	74.9	77.5	79.7	80.7	82.0	82.9
	RS	2.6	3.7	3.8	4.0	4.1	4.1	4.1	4.1	4.1	4.1
	Visual	30.5	47.1	56.5	62.6	67.3	70.7	72.9	75.2	76.8	78.7

In addition to ranking the candidate characters directly with their ENOPs, we also ranked the characters with their PMIs, shown in Equation (2) and Equation (4) in Section 5.1, and repeated the experiments with Elist and Jlist. The lower part of Table XIII shows the observed statistics. Qualitatively, the statistics in the upper and the lower parts of Table XIII do not show different trends: SS and SC3 remain to be the most effective methods to find phonologically and visually similar errors. Overall, finding phonologically similar errors is easier than finding visually similar errors. Providing candidate lists that had only 10 characters achieved reasonable performances.

The most noticeable difference between the upper and the lower part of Table XIII is in the R_1 column. It appears that, if we would recommend only one candidate character, using the raw values of ENOPs to rank will offer better inclusion rates than using PMI. Although this observation appears to be appropriate for the statistics in Table XIII that we collected from the experiments that used Jlist, this trend did not survive in our experiments with Elist.

Table XIV shows exactly the same sets of statistics as those shown in Table XIII. The only difference is that we used Elist, rather than Jlist, to repeat all of the experiments that we used to obtain Table XIII. Statistics in Table XIV indicate the same trends as those suggested by the most of statistics in Table XIII, so we do not repeat the same statements.

A major contribution of the statistics in Table XIV appears in Figure 4, which shows that using PMIs or ENOPs did not guarantee that there would be differences in the performances. We drew the left chart based on the data in Table XIII and the right chart based on data in Table XIV. The curves in the left chart show that using PMIs offered inferior performance than using the raw values of ENOPs, and the curves in the right chart show the opposite trend.

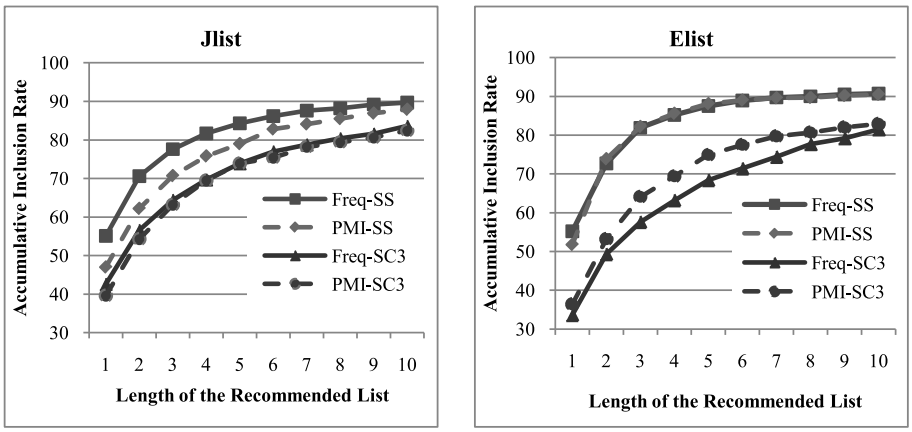


Fig. 4. Using PMI does not necessarily outperform using ENOPs.

Table XV. Ranking the Candidates Based on ENOPs and PMIs for Ilist

	Frequencies					PMIs				
	R_1	R_2	R_3	R_4	R_5	R_1	R_2	R_3	R_4	R_5
SS	70.3	77.7	80.6	81.0	81.6	66.7	76.0	80.2	81.0	81.6
SD	25.6	28.3	28.9	28.9	29.3	24.8	28.1	28.9	29.1	29.3
MS	1.4	1.7	1.7	1.7	1.7	1.6	2.1	2.1	2.1	2.1
MD	1.6	1.6	1.6	1.6	1.6	1.2	1.6	1.6	1.6	1.6
Phone	76.7	89.1	93.6	94.8	95.5	70.5	86.2	92.1	94.4	95.7
SC1	64.7	72.0	76.0	78.0	78.3	61.7	72.3	75.0	76.7	77.7
SC2	58.0	64.7	68.3	70.7	71.0	53.7	66.3	69.3	70.3	70.3
SC3	71.3	80.0	85.0	86.7	87.0	67.0	80.0	84.7	86.0	86.3
RS	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3
Visual	71.0	82.3	86.3	89.3	89.3	65.0	81.7	86.7	88.3	89.0

Although PMIs are frequently used to compute the co-occurrences of two events, including the collocation of words, examining Formula (4) discussed in Section 5.1 reveals an intuitive interpretation of the PMIs in our applications. The formula measures the percentages of observing the incorrect words (i.e., $C \wedge X$) given that the candidate characters appeared (i.e., X in the formula; recall that this is the character that would replace the correct character). Such percentages can be diluted if X happens to represent a high frequent character.

Similar to an experiment that we conducted for the Jlist, we also created the union list SS + SD for the experiments with Elist, and the results are shown in the middle of Table XIV. This time, the SS + SD list outperformed the Phone list by a margin by about 5%. However, the resulting performance profile of SS + SD is still inferior to that of SS. Interestingly, when we consider the top 10 candidate characters, the SS + SD outperformed not only the Phone but also the SS list marginally.

5.6 Ranking Tests with Ilist

Table XV shows the accumulative inclusion rates (AIRs) for the experiments for Ilist, which provides reported errors for simplified Chinese. The inclusion rates for the experiments for Ilist were presented in Table XI.

Table XVI. AIRs for Ranking Candidates for Wlist Based on Frequencies and PMIs

		R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
Frequency	SS	73.1	84.6	90.8	92.3	93.1	93.8	93.8	93.8	93.8	93.8
	SD	18.5	21.5	22.3	23.1	23.1	23.1	23.1	23.1	23.1	23.1
	MS	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
	MD	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
	Phone	68.5	82.3	88.5	91.5	93.1	93.1	95.4	95.4	96.2	96.2
	SC1	57.3	70.9	74.8	77.7	78.6	78.6	78.6	79.6	79.6	79.6
	SC2	54.4	69.9	72.8	73.8	74.8	76.7	76.7	77.7	77.7	78.6
	SC3	60.2	76.7	82.5	84.5	85.4	86.4	89.3	89.3	89.3	89.3
	RS	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Visual	52.4	67.0	77.7	81.6	85.4	86.4	86.4	86.4	89.3	89.3
PMI	SS	63.8	83.1	90.0	91.5	92.3	92.3	92.3	92.3	92.3	92.3
	SD	18.5	21.5	22.3	22.3	23.1	23.1	23.1	23.1	23.1	23.1
	MS	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
	MD	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
	Phone	60.8	82.3	90.8	92.3	94.6	96.2	96.2	96.2	96.2	96.9
	SC1	51.5	64.1	71.8	73.8	77.7	78.6	79.6	79.6	79.6	79.6
	SC2	49.5	65.0	71.8	72.8	75.7	75.7	77.7	77.7	77.7	77.7
	SC3	55.3	73.8	80.6	85.4	88.3	89.3	89.3	89.3	89.3	89.3
	RS	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Visual	52.4	67.0	71.8	77.7	84.5	86.4	86.4	87.4	88.3	88.3

We use a format that is similar to the format for Tables XIII and XIV for Table XV, but we list only the AIRs for the top-five candidates. When we used the top-five candidate characters, the AIRs were almost as good as the inclusion rates, which we listed in Table XI. Statistics in Table XV indicate that our system performed well for the simplified Chinese. We can find results similar to those that we presented for the experimental results for Elist and Jlist. The candidate lists selected with the SS and SC3 criteria were the most effective in capturing phonologically and visually related errors. SD continued to serve as an instrumental complement for SS.

In contrast to what we observed in the experiments for Elist and Jlist, the Phone lists, which are the unions of SS, SD, MS, and MD lists, performed better than the best performing individual lists, that is, SS lists. The top-five candidate characters in the Phone lists captured 95.5% of the phonologically related errors on average. Analogously, the Visual list, which is the union of SC1, SC2, SC3, and RS, captured 89.3% of the visually related errors.

5.7 Further Tests with Wlist and Blist

We explained, in Section 4.2, that we used Wlist and Blist as the new datasets to test how our system will perform with unforeseen data.

Table XVI and Table XVII provided in the Appendix show that the experimental results for Wlist and Blist were not different from the results for Elist and Jlist, which we discussed in Section 5.5. The inclusion rates were good, as we discussed in Section 5.4. Using the top 10 candidate characters enabled us to catch most of the errors that we were able to capture with the complete lists. Using PMI and ENOPs to rank the candidate characters achieved performance profiles of similar quality. SS lists and SC3 lists performed best if we had to use only one of the lists to capture the phonologically and the visually related errors, respectively. In addition, SD lists complemented the SS lists to find those phonologically related errors.

Table XVII. AIRs for Ranking the Candidates for Blist Based on Frequencies and PMIs

		R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}
Frequency	SS	69.1	78.3	80.3	80.9	81.5	81.8	81.8	81.8	81.8	81.8
	SD	20.1	22.0	22.9	23.9	23.9	23.9	23.9	23.9	23.9	23.9
	MS	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
	MD	1.6	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9	1.9
	Phone	71.3	82.5	86.3	89.2	90.4	92.4	93.0	93.9	93.9	93.9
	SC1	64.2	71.6	76.1	76.9	77.6	77.6	77.6	77.6	77.6	77.6
	SC2	59.7	67.2	70.1	72.4	73.9	74.6	74.6	74.6	74.6	74.6
	SC3	75.4	85.1	87.3	88.1	88.8	88.8	89.6	89.6	90.3	90.3
	RS	3.0	3.7	3.7	3.7	3.7	3.7	3.7	3.7	3.7	3.7
	Visual	71.6	79.9	83.6	86.6	88.1	89.6	89.6	89.6	89.6	89.6
PMI	SS	64.3	75.8	77.4	80.3	80.9	81.2	81.5	81.5	81.5	81.8
	SD	19.7	22.3	22.6	23.9	23.9	23.9	23.9	23.9	23.9	23.9
	MS	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2	3.2
	MD	1.6	1.6	1.6	1.6	1.6	1.9	1.9	1.9	1.9	1.9
	Phone	67.5	83.4	86.9	89.5	90.8	92.4	93.0	93.0	93.0	93.0
	SC1	63.4	73.1	76.9	77.6	77.6	77.6	77.6	77.6	77.6	77.6
	SC2	61.9	70.9	73.9	73.9	74.6	74.6	74.6	74.6	74.6	74.6
	SC3	75.4	83.6	87.3	89.6	89.6	89.6	89.6	89.6	89.6	89.6
	RS	3.7	3.7	3.7	3.7	3.7	3.7	3.7	3.7	3.7	3.7
	Visual	74.6	82.8	87.3	89.6	90.3	91.0	91.0	91.0	91.0	91.0

6. APPLICATIONS

With the capability to capture the actual errors that occurred while people typed and wrote Chinese, we can apply our techniques to computer assisted language learning and to the related fields that we mentioned in Section 1.

The most obvious application is to help teachers prepare test items for "Incorrect Character Correction" tests (ICC tests). In such tests, students have to find and correct an incorrect Chinese character in a given sentence, for example, "小明昨天參加了校外施行" (Ming took part in the field trip yesterday. Xiao3 ming2 zuo2 tian1 can1 jia1 le1 xiao4 wai4 shi1 xing1). In this Chinese string, "施" (shi1) is incorrect and should be changed to "旅" (lu3) to make the statement correct. This is a very common type of test in the assessment of Chinese language proficiency.

To prepare such test items, there may be some characters which teachers wish to check if their students recognize in their correct form or not, that is, "旅" in "旅行" (lu3 xing2) in the previous example. When preparing the test items, teachers will have to figure out what might be the most appropriate incorrect character to use to stand in for the correct character. Depending on the level of difficulty and the purpose of the test, they may prefer incorrect characters that are visually similar or are phonologically similar to the correct character. For phonologically similar characters, teachers may prefer to select incorrect characters that were recommended with the SS, SD, MS, and MD criteria. From this viewpoint, we should present candidate characters by their categories. The union lists are not the best choice.

Figure 5 shows a snapshot of the user interface of our prototype[16] that aims to help teachers prepare test items for ICC tests. In this example, a teacher requested errors that were visually similar ("形體相似" in the figure; xing2 ti3 xiang1 si4) and errors

[16]See http://140.119.164.139/biansz/bianszindex_v2.php. This is our own service and it is always open, except when we experience power outage problems.

Fig. 5. The interface of our prototype for assisting teachers to prepare test items in the "Incorrect Character Correction" tests.

that had the same sound and same tone ("同音同調" in the figure; tong2 yin1 tong2 diao4), and our system returned only the top three candidates. This is a conservative design, given that we are able to capture a large percentage of the actual incorrect characters of previously observed errors with the top 10 candidates (Sections 5.5 to 5.7).

This authoring tool can be evaluated by how often the recommended characters are adopted by teachers in their test items. This style of evaluation is the same as what we have accomplished in Section 5.4 through Section 5.7. The correct words in the lists, in Table VI, serve as the target test items, and the actual incorrect characters are the teachers' choices. From this viewpoint, we have conducted an evaluation with more than 4,100 individual tests. The observed inclusion rates, which were presented in Table XI, show that our system was able to offer candidate characters that included the teachers' choices. The accumulate inclusion rates, which were presented from Table XIII to Table XVII, further indicate that, by providing no more than 10 candidate characters in different categories of similar characters, our system maintained its efficacy for assisting the compilation of test items for ICC tests.

In addition to assisting the preparation of test items for Chinese tests, we can employ the lists of similar characters to automatic detection of errors in Chinese text (e.g., Zhang et al. [2000]). The statistics discussed in Section 4.3 show that a large portion of errors in Chinese texts are related to characters that have the same or similar pronunciations, and a previous work applied phonetic information for this error detection task based on related arguments [Huang et al. 2008]. Using both visually and phonetically similar characters along with statistical methods, we significantly improve the performances of Huang et al.'s system and two other systems that were reported in the literature [Wu et al. 2010].

We plan to offer a free and open Web-based service to the research community. The service will allow users to enter queries to search Chinese characters that meet certain conditions. With a minor change of the interface shown in Figure 5, we can offer psycholinguistic researchers the neighbor words (e.g., Lo and Hue [2008], Tsai et al. [2006]) of a given Chinese word for their studies. In fact, we are applying our system to support the design of educational games for cognition-based learning of Chinese characters (cf., Lee et al. [2010]). Moreover, our work can be used to find the suggested queries when users of search engines enter incorrect words [Croft et al. 2010, p. 197]. Although we are not experts in the recognition of Chinese characters either in printed

(i.e., OCR) or in written form, we can help researchers to find the confusion sets for Chinese characters [Fan et al. 1995] more efficiently.

7. DISCUSSIONS

The experimental results presented in Section 5 showed that it is relatively easy to capture errors that are related to phonological similarity. It is relatively harder to catch errors that are related to visual similarity.

Using the information about the pronunciations of characters that are available in Chinese lexicons is very effective for reproducing phonologically-related errors. Selecting candidate characters with the SS and SD criteria was most fruitful. The main reason that our program did not catch the errors that were related to phonological similarity was that our lists of confusing phonemes (cf. Table I) did not contain the types of errors that actually occurred. This can happen if the types of errors are those which are not considered significant in psycholinguistic studies, but which can occur once in a while in reality.

Using the extended Cangjie codes proved to be the main reason why we could capture a larger portion of the errors that are related to visual similarity, when we compare the performances of our systems that were implemented in 2007 and 2008. The decision to divide characters into subareas further improves our ability to find similar characters. However, the steps demand subjective decisions, in which we observed how characters were divided [Lee 2010a], and these decisions will influence how well we find the incorrect characters. We discussed an example of the problem, that is, the "弓 人一" (gong1 ren2 yi1) problem, in Section 3.4. Another example is the question of how our programs may find the similarity between "副" (fu4) and "福" (fu2). According to Lee [2010a], the LIDs for "副" and "福" are 4 and 3, respectively (cf., Section 3.3). Hence, the shared component "畐" (fu2) of these two characters will be saved in two different ways: "一 口 田" (yi1 kou2 tian1) at P1 for "副"; and "一 口" and "田" at P1 and P2, respectively, for "福".

To alleviate the problem, we concatenated the substrings of Cangjie codes into one string and computed the Dice coefficient (Equation (1) in Section 3.4) of the concatenated Cangjie codes for two characters. This strategy proved to be very important. Using SC3 to select visually similar characters outperformed SC2 and SC1 in all of our experiments.

Although we have achieved good experimental results, using Cangjie codes as the basis of defining the visual similarity between characters does not produce perfect results. The original Cangjie codes may not reflect the complexity, for example, the number of strokes, of a component in a character. A complex component can be represented with a simple Cangjie symbol, for example, the Cangjie code for "弗" (fu2) is "中 中 弓" (zhong1 zhong1 gong1). In contrast, a seemingly simple component can be represented with a longer sequence of Cangjie symbols, for example, the complete Cangjie code for "予" (yu3) is "弓 戈 弓 弓" (gong1 ge1 gong1 gong1). This phenomenon may mislead our score functions, that is, SC1, SC2, and SC3, which rely on the lengths of the matched Cangjie codes to determine the degree of similarity between characters.

A possible solution to this problem is to use our own Cangjie codes for the basic elements, but this strategy has its problems. For instance, we replaced the Cangjie code for "言" (yan2) with "卜 一 一 口" (bu3 yi1 yi1 kou3) in c_5, c_6, and c_7 in Table IV. However, such an operation is extremely subjective and labor intensive. Although we changed the Cangjie codes for a limited number of elements, we cannot guarantee that we have done enough for all possible errors that are related to visual similarity. Moreover, we were not sure whether we were just trying to maximize the performance of our systems in the case of some particular lists of errors. This was the main reason

that we collected Wlist and Blist for further experiments, after we had been using Elist and Jlist for an extended period of time. Fortunately, the experimental results for Wlist and Blist remained satisfactory.

Another problem that came up when we built the database of the extended Cangjie codes is the degree of detail to which we should recover the Cangjie codes. Consider this list of characters: "舞" (wu3), "列" (lie4), "例" (li4), "夥" (huo3), and "麥" (mai4). It is probably not easy for everyone to notice that they all share "夕" (xi4) somewhere inside them. To what degree do users pay attention to relatively small elements? Should we consider this factor when we design the score functions to measure the degree of similarity between two characters? This is a hard question for us. The best design may depend on the actual applications, for example, the needs of psycholinguistic experiments [Leck et al. 1995; Yeh and Li 2002].

So far, we have not touched upon the issue that the Cangjie codes do not provide a good mechanism for comparing the similarities between characters that consist of very few strokes. Examples are c_1(田, tian2), c_2(由, you2), c_3(甲, jia3), and c_4(申, shen1) in Table II. Another group of similar characters are "土" (tu3), "士" (shi4), "工" (gong1), "干" (gan1), and "千" (qian1). Differences among these characters are at the stroke level, so we cannot rely on the Cangjie codes to find their similarities. For such characters, the Wubihua encoding method [Wubihua 2010] should be applied. The Wubihua encoding method assigns identification numbers to a selected set of strokes, for example, "1" for horizontal strokes and "2" for vertical strokes. Because there is exactly one canonical way to write a Chinese character, that is, the standard order of the strokes that form the character, one can convert each of the strokes into their Wubihua digits, and use this sequence of digits to encode a Chinese character. The Wubihua codes for "土", "士", "工", "干", and "千" are, respectively, "121", "121", "121", "112" and "312"; and the Wubihua codes for "田", "由", "甲", and "申" are, respectively, "25121", "25121", "25112", and "25112". Demanding an exact match between strings as the selection criterion, we can find that "土", "士", and "工" are more similar to each other than to "干" and "千". By appropriately integrating the extended Cangjie codes and Wubihua codes, we will be able to extend our ability to find visually similar characters to a larger scope of characters.

For the study of incorrect Chinese characters, we have intentionally put aside an important class of errors at this moment. For written characters, people may write incorrect characters that look like correct characters, for example, writing "試" (shi4) as "試".[17] These so-called pseudo-characters obey the formation principles of Chinese characters, but, in fact, do not belong to the language. These incorrect characters were not considered in the current study because we could not normally enter them into our files as they were not contained in the font files. Nevertheless, studying this type of errors may uncover possible ways that people memorize Chinese characters, and opens another door to the mental lexicons of Chinese learners.

Song and his colleagues propose methods for automatic proofreading for simplified Chinese in Song et al. [2008]. They consider seven operators for building Chinese characters from their components and propose a set of rules for computing the similarity between Chinese characters. They then employ the similar characters with statistical information about language models to detect possible incorrect words. This line of work is very similar to the work presented in this article. It will be very interesting to compare the performances of Song et al.'s and our systems with some common test sets.

SJTUD [1988] provides not just a systematic way to decompose simplified Chinese characters, and it also lists the decompositions of 11,254 individual characters. It

[17]Reported in the *United Daily News* (http://www.udn.com.tw) on 19 May 2010.

will be very interesting to compare the effectiveness for computing visually similar characters with the extended Cangjie codes and the decompositions in SJTUD [1988].

8. CONCLUSIONS

We found methods to reproduce the errors found in the writing of Chinese script. The methods utilized information about the pronunciation of Chinese characters and the heuristics rules that were derived from observations in psycholinguistic studies to judge the degree of similarity between pronunciations. The methods also employed the extended Cangjie codes and score functions to determine the degree of visual similarity between characters.

We evaluated our approach from three aspects. In Section 5.3, we compared the Web-based statistics to show the reliability of Web-based statistics. In Section 5.4, we showed that our approach could capture the incorrect character for a diverse scope of test data at satisfactory rates. In Sections 5.5 through 5.7, we applied and compared two different methods to rank the candidate characters in an attempt to capture the incorrect character with shorter lists of candidate characters. The experiments were carried out with data that we presented in previous conference articles and some new data that covered both traditional and simplified Chinese.

In these experiments, it was found that 76% of these errors were related to phonological similarity and that 46% were related to visual similarity between characters. We showed that the Web-based statistics were reasonably stable when we compared the popularity of word usages by comparing the numbers of Web pages that contained the target words in both 2009 and 2010. Experimental results show that we were able to capture 97% of the 4,100 errors, when we recommended 104 candidate characters. When we recommended only 10 candidate characters, we still caught more than 80% of the 4100 errors. The reported techniques are useful for applications that are related to Chinese, and, in particular, we showed a real-world application that can help teachers to author test items for "incorrect character correction" tests for Chinese.

ACKNOWLEDGMENTS

We thank anonymous reviewers of this journal version and the previous conference articles for their invaluable comments, which strongly influenced this publication. Experiments were added and improved to respond to reviewers' comments, though we did not mark each of such experiments to indicate the reviewers' credits. We would also like to thank Professor Song Rou for his sharing his article with us. We would also like to thank Miss Moira Breen for her indispensable support for our English.

REFERENCES

CANGJIE. 2010. An introduction to the Cangjie input method. http://en.wikipedia.org/wiki/Cangjie_input_method.

CDL. 2010. Chinese document laboratory, Academia Sinica. http://cdp.sinica.edu.tw/cdphanzi/. (In Chinese)

CHEN, M. Y. 2000. *Tone Sandhi: Patterns Across Chinese Dialects* (Cambridge Studies in Linguistics 92). Cambridge University Press.

CHU, B.-F. 2010. *Handbook of the Fifth Generation of the Cangjie Input Method*. http://www.cbflabs.com/book/5cjbook/. (In Chinese)

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2009. *Introduction to Algorithms* 3rd Ed. MIT Press.

CROFT, W. B., METZLER, D., AND STROHMAN, T. 2010. *Search Engines: Information Retrieval in Practice*. Pearson.

DICT. 2010. An official source of information about traditional Chinese characters. http://www.cns11643.gov.tw/AIDB/welcome.do.

FAN, K.-C., LIN, C.-K., AND CHOU, K.-S. 1995. Confusion set recognition of online Chinese characters by artificial intelligence technique. *Patt. Recog. 28*, 3, 303–313.

FELDMAN, L. B. AND SIOK, W. W. T. 1999. Semantic radicals contribute to the visual identification of Chinese characters. *J. Mem. Lang. 40*, 4, 559–576.

FROMKIN, V., RODMAN, R., AND HYAMS, N. 2002. *An Introduction to Language* 7th Ed. Thomson.

HANDICT. 2010. A source for traditional and simplified Chinese characters. http://www.zdic.net/appendix/f19.htm.

HUANG, C.-M., WU, M.-C., AND CHANG C.-C. 2008. Error detection and correction based on Chinese phonemic alphabet in Chinese text. *Int. J. Uncertainty, Fuzziness Knowl.-Base. Syst. 16*, suppl. 1, 89–105.

JACKENDOFF, R. 1995. *Patterns in the Mind: Language and Human Nature*. Basic Books.

JUANG, D., WANG, J.-H., LAI, C.-Y., HSIEH, C.-C., CHIEN, L.-F., AND HO, J.-M. 2005. Resolving the unencoded character problem for Chinese digital libraries. In *Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries (JCDL'05)*. 311–319.

JURAFSKY, D. AND MARTIN, J. H. 2009. *Speech and Language Processing* 2nd Ed. Pearson.

KUO, W.-J., YEN, T.-C., LEE, J.-R., CHEN, L.-F., LEE, P.-L., CHEN, S.-S., HO, L.-T., HUNG, D. L., TZENG, O. J.-L., AND HSIEH, J.-C. 2004. Orthographic and phonological processing of Chinese characters: An fMRI study. *NeuroImage 21*, 4, 1721–1731.

LECK, K.-J., WEEKES, B. S., AND CHEN, M.-J. 1995. Visual and phonological pathways to the lexicon: Evidence from Chinese readers. *Mem. Cogn. 23*, 4, 468–476.

LEE, C.-Y. 2009. The cognitive and neural basis for learning to reading Chinese. *J. Basic Educ. 18*, 2, 63–85.

LEE, C.-Y., HUANG, H.-W., KUO, W.-J., TSAI, J.-L., AND TZENG, O. J.-L. 2010. Cognitive and neural basis of the consistency and lexicality effects in reading Chinese. *J. Neurolinguist. 23*, 1, 10–27.

LEE, C.-Y., TSAI, J.-L., HUANG, H.-W., HUNG, D. L., AND TZENG, O. J.-L. 2006. The temporal signatures of semantic and phonological activations for Chinese sublexical processing: An event-related potential study. *Brain Res. 1121*, 1, 150–159.

LEE, H. 2010a. Cangjie Input Methods in 30 Days 2. Foruto. http://input.foruto.com/cccls/cjzd.html. (In Chinese)

LEE, MU. 2010b. A quantitative study of the formation of Chinese characters. http://chinese.exponode.com/0_1.htm. (In Chinese)

LIU, C.-L., LAI, M.-H., CHUANG, Y.-H., AND LEE, C.-Y. 2010. Visually and phonologically similar characters in incorrect simplified Chinese words. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*. 739–747.

LIU, C.-L., LEE, C.-Y., TSAI, J.-L., AND LEE, C.-L. 2011. Forthcoming. A cognition-based interactive game platform for learning Chinese characters. In *Proceedings of the 26th ACM Symposium on Applied Computing (SAC'11)*.

LIU, C.-L. AND LIN, J.-H. 2008. Using structural information for identifying similar Chinese characters. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08)*. 93–96.

LIU, C.-L., TIEN, K.-W., CHUANG, Y.-H., HUANG, C.-B., AND WENG, J.-Y. 2009a. Two applications of lexical information to computer-assisted item authoring for elementary Chinese. In *Proceedings of the 22nd International Conference on Industrial Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE'09)*. 470–480.

LIU, C.-L., TIEN, K.-W., LAI, M.-H., CHUANG, Y.-H., AND WU, S.-H. 2009b. Capturing errors in written Chinese words. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL'09)*. 25–28.

LIU, C.-L., TIEN, K.-W., LAI, M.-H., CHUANG, Y.-H., AND WU, S.-H. 2009c. Phonological and logographic influences on errors in written Chinese words. In *Proceedings of the 7th Workshop on Asian Language Resources (ALR'09)*. 84–91.

LIU, C.-L., JAEGER, S., AND NAKAGAWA, M. 2004. Online recognition of Chinese characters: The state-of-the-art. *IEEE Trans. Patt. Anal. Mach. Intel. 26*, 2, 198–213.

LO, M. AND HUE, C.-W. 2008. C-CAT: A computer software used to analyze and select Chinese characters and character components for psychological research. *Behav. Res. Meth. 40*, 4, 1098–1105.

MA, W.-Y. AND CHEN, K.-J. 2003. Introduction to CKIP Chinese word segmentation system for the first international Chinese word segmentation bakeoff. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing (SIGHAN'03)*. 168–171.

MANNING, C. D. AND SCHÜTZE, H. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

MOE 1996. Common errors in Chinese writings (常用國字辨似). Ministry of Education, Taiwan. http://140.111.34.54/files/site_content/M0001/biansz/c9.htm.

SISON, R. AND SHIMURA, M. 1998. Student modeling and machine learning. *Int. J. Artif. Intell. Educ. 9*, 1, 128–158.

SJTUD. 1988. Chinese character code workgroup of Shanghai Jiao Tong University. In *A Dictionary of Chinese Character Information*. Beijing Science Press (in Chinese).

SONG, R., LIN, M., AND GE, S.-L. 2008. Similarity calculation of Chinese character glyph and its application in computer aided proofreading system. *J. Chin. Comput. Syst. 29*, 10, 1964–1968. In Chinese.

SUN, X., CHEN, H., YANG, L., AND TANG, Y. Y. 2002. Mathematical representation of a Chinese character and its applications. *Int. J. Patt. Recog. Artif. Intell. 16*, 6, 735–747.

TSAI, J.-L., LEE, C.-Y., LIN, Y.-C., TZENG, O. J.-L., AND HUNG, D. L. 2006. Neighborhood size effects of Chinese words in lexical decision and reading. *Lang. Linguist. 7*, 3, 659–675.

TSAY, Y.-C. AND TSAY, C.-C. 2003. *Diagnoses of Incorrect Chinese Usage* (In Chinese). Firefly Publisher.

UNICODE. 2010. Unicode Standard 4.0.1, Chapter 11. http://www.unicode.org/versions/Unicode4.0.0/ch11.pdf.

VIRVOU, M., MARAS, D., AND TSIRIGA, V. 2000. Student modelling in an intelligent tutoring for the passive voice of English language. *Educ. Technol. Soc. 3*, 4, 139–150.

WU, S.-H., CHEN, Y.-Z., YANG, P.-C., KU, T., AND LIU, C.-L. 2010. Reducing the false alarm rate of Chinese character error detection and correction. In *Proceedings of the 1st Joint Conference on Chinese Language Processing (SIGHAN'10)*. 54–61.

WUBIHUA. 2010. An input method used in mainland China. http://en.wikipedia.org/wiki/Wubihua_method.

YEH, S.-L. AND LI, J.-L. 2002. Role of structure and component in judgments of visual similarity of Chinese characters. *J. Experi. Psych. Hum. Percept. Perform. 28*, 4, 933–947.

ZHANG, L., ZHOU, M., HUANG, C., AND PAN, H. 2000. Automatic detecting/correcting errors in Chinese text by an approximate word-matching algorithm. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*. 248–254.

ZIEGLER, J. C., TAN, L. H., PERRY, C., AND MONTANT, M. 2000. Phonology matters: The phonological frequency effect in written Chinese. *Psychol. Sci. 11*, 3, 234–238.

Received September 2010; revised December 2010; accepted January 2011