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 interpret 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, 珎 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 珎 (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 standards, 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	拾得奇珎異宝，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.

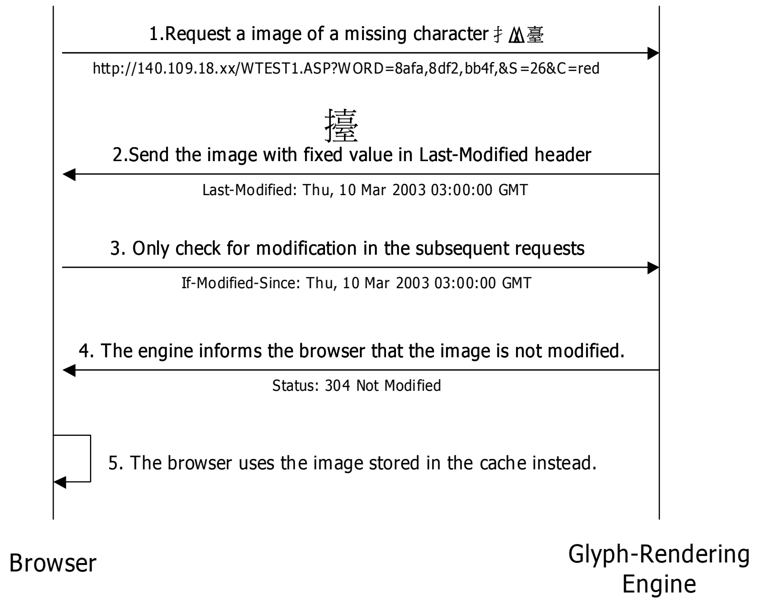
6.1.2 Bandwidth Efficiency

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


1.Request a image of a missing character 扌从臺
http://140.109.18.xx/WTEST1.ASP?WORD=8afa,8df2,bb4f,&S=26&C=red
擡
2.Send the image with fixed value in Last-Modified header
Last-Modified: Thu, 10 Mar 2003 03:00:00 GMT
3. Only check for modification in the subsequent requests
If-Modified-Since: Thu, 10 Mar 2003 03:00:00 GMT
4. The engine informs the browser that the image is not modified.
Status: 304 Not Modified
5. The browser uses the image stored in the cache instead.
Browser
Glyph-Rendering Engine

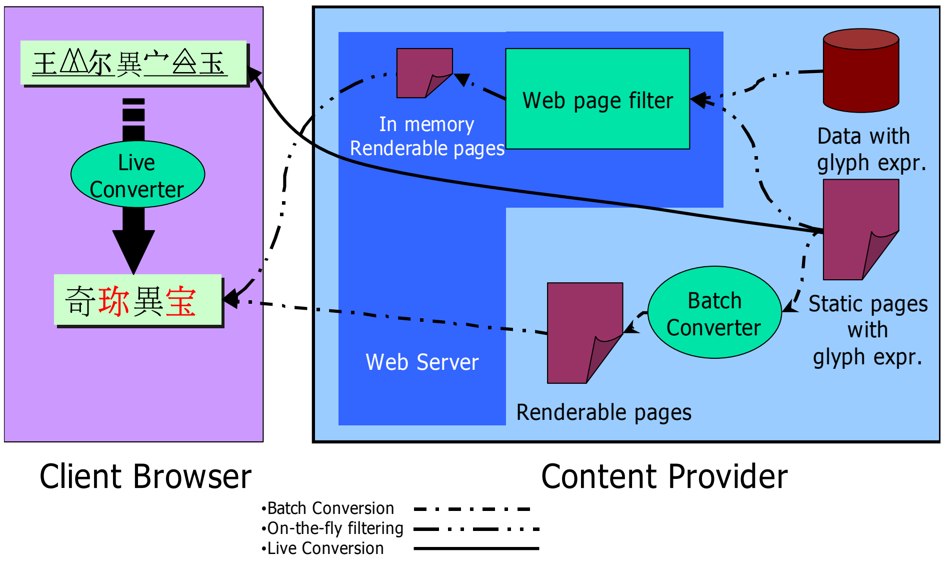
6.1.3 Browser Compatibility

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 highquality 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 LiveConnect (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