Does Learning How to Read Japanese Have to Be So Difficult, And Can the Web Help?

Julien Quint
currently unaffiliated; formerly of
National Institute of Informatics, Tōkyō, Japan
julien.quint@imag.fr

Ulrich Apel
currently unaffiliated; formerly of
National Institute of Informatics, Tōkyō, Japan
ulrich_apel@t-online.de

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Dictionaries; H.3.5 [**Online Information Services**]: Web-based; I.2.7 [**Natural Language Processing**]: Text analysis—*Automatic transliteration*

General Terms

Algorithms, Languages

Keywords

Japanese, reading help, *kanji*, graphetic dictionary, SVG

1. INTRODUCTION

A major roadblock that learners of the Japanese language encounter is the complex writing system which uses two different syllabaries (*hiragana* and *katakana*) in addition to thousands of characters originally borrowed from China (the *kanji*.) These characters can be very complex graphically and usually have several pronunciations (named *onyomi* and *kunyomi*, the first being derived from Chinese.) Achieving full literacy in Japanese means knowing more than 2,000 *kanji*–an intimidating prospect. Moreover, learning *kanji* traditionally takes years of education (from grade school through high school) and relies mostly on rote memorization, through incessant homework and tests. Such a process, and by extension the related learning material, are inadequate for most foreign learners.

Our goal is to provide tools to help overcome literacy problems, by means of a twofold approach. The first aspect is *automatic transliteration* of Japanese text, allowing an easier way to read electronic text. The second aspect is a *graphetic kanji dictionary* which describes every *kanji* in terms of its graphical components (using XML and SVG), allowing for many interesting applications[1].

2. AUTOMATIC TRANSLITERATION

Transliteration is a way to make written text more accessible. By removing the *kanji* from the picture, a large roadblock disappears. Two questions then appear: what should the *kanji* be replaced with, and how?

If the learner is a complete beginner, or if a tourist needs to read the name of a place and does not know anything about Japanese writing, the answer is to *romanize* the text, *i.e.* rewriting it completely with only roman letters. Making a little progress, we can read the *hiragana* and *katakana* and thus render the text phonetically in either syllabary. And if we are serious about learning *kanji* but still need some help, we can keep the text as is but add pronunciation information to the kanji in *hiragana* (these are called *furigana* or *ruby*.)

For example, take the following sentence: "私は京都へ行きたい。" ("I want to go to Kyōto.") One romanization[2] of this sentence is: "*watashi wa Kyōto e ikitai*." In *hiragana*, we introduce whitespace and replace all *kanji* by their pronunciation, which leads to: "わたし は きょうと へ いきたい。" Lastly, keeping the *kanji* but adding *furigana*, we get: "私(わたし) は 京都(きょうと)へ行(い)きたい。"

The automatic transliteration itself is an interesting question as it necessitates first to segment the text into words (spacing is rare in Japanese writing, which is just another difficulty) and lookup the correct pronunciation of the identified word in some dictionary. Our solution is based on the MeCab morphological analyzer[3].

The process is divided in two steps. First step is to feed the input text to MeCab, which returns a sequence of tokens with their surface (original) form, part-of-speech and pronunciation. In the second step, if the desired output is *furigana*-annotated text or *hiragana*/*katakana* text, we simply replace the surface form of all tokens with their pronunciation (adding white space between tokens for readability.) In the case of romanized text, we go further and rewrite all pronunciation strings using a *transliteration table* for the selected romanization style. The tables map *hiragana* combinations (*e.g.* か or ちゅ) to their transliteration (*ka* or *chu*.)

3. GRAPHETIC DESCRIPTION OF KANJI

In the previous section, we tried to get rid of *kanji*. Here, the goal is to look at them in as much detail as possible in order to produce better teaching material: knowing how to write *kanji* is arguably the best way to really memorize them. This is achieved by considering kanji as hierarchical, graphical entities, and representing the corresponding data in XML and SVG[4] for the visual part.

[1]A more detailed presentation of this data is available at http://www.svgopen.org/2004/papers/svgopen/

[2]There are actually several romanizations style in use for Japanese, often inconsistently.

[3]http://chasen.org/~taku/software/mecab/

[4]http://www.w3.org/Graphics/SVG/

WWW 2005, May 10–14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

The smallest meaningful unit out of which a *kanji* can be described is the stroke (originally, the strokes of the brush.) Our analysis of strokes distinguishes 25 basic forms of strokes, considering stroke direction, bending, endings (blunt or with a short bend), and so on.

There is an intermediary unit between the stroke and the full character. In dictionaries, one finds *radicals*, which are groups of strokes giving semantic or phonetic values to the character that they are part of. There are actually many more such groups, but surprisingly there does not seem to exist a technical term in Japanese for them. We choose to call them *grapheme elements*, or more simply, *components*.

As a very simple example, the character 明 ("bright") is made of two components, 日 on the left side, and 月 on the right side. These two components, in turn, are made of strokes. The strokes identified during the graphetic analysis of the characters have been numbered, so we can represent our character as two groups of strokes, *i.e.* (3, 11, 2, 2) and (4, 17, 2, 2). Since a stroke of a given type is always drawn in the same direction this gives us exact information about the correct drawing of this *kanji*. More complex characters have components that can be divided into further components.

This information is represented as an XML file for every character, but the actual graphical data is represented in a parallel SVG file, where strokes are represented as paths and components as groups. The graphical part of the data was built over a year using Adobe Illustrator and exported into SVG. The result of this work is a database of approximately 6,500 *kanji*.

4. AAAA: A WEB-BASED INTERFACE

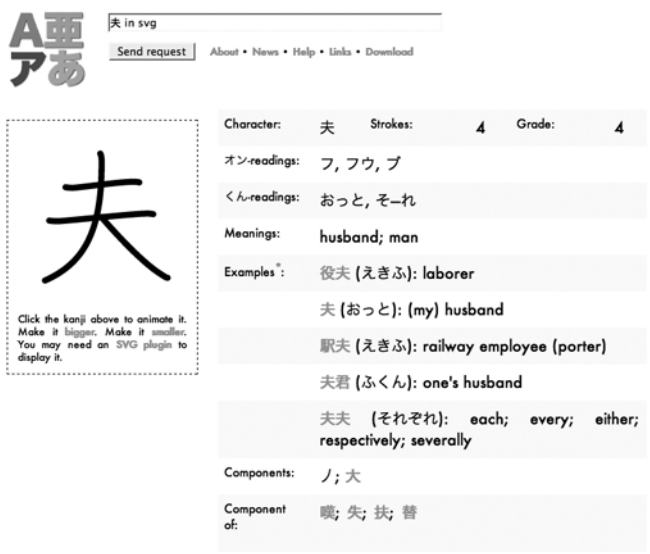

Figure 1: Displaying information about a character. On the left side, the SVG display can be clicked to show the drawing of the character. On the right side, dictionary informations are displayed (number of strokes, readings, example words generated randomly), as well as the related characters (which are components of this character, or which this character is a component of.)

In this third part, all the work presented in this paper comes together. There are already many transliteration services that exist on the web, such as Kakasi[5]; and there also exist many different electronic dictionaries for Kanji such as Jim Breen's Kanjidic[6].

Our objective is to provide a unified platform for applications related to transliteration; namely transliterating text inline or on the Web and getting information about *kanji* found in electronic text. The unification of these tasks is made possible by a very simple request language, recalling works such as Google's calculator feature[7], based on a flexible natural-language interface.

Figure 1 shows the interface of the Aaaa system, currently accessible through a public test server[8]. The form on top is used to input *transliteration requests* using a simple request language. An example request is "漢字 `in ascii`": here, the user wants the romanization (using only 7-bit ASCII) of a string, which results in "`kanji`". We can also transliterate complete web pages by inputting an URL instead of text: "`http://www.asahi.com/ with furigana`" rewrites the web page at the given URL with Ruby annotations[9].

Looking up the *kanji* dictionary uses the same interface. The requests are of the form "漢字 `in SVG`"; from the result page, individual *kanji* available in the dictionary can be accessed. The main part is the SVG window which can be clicked to show an animation of the drawing of the character, illustrating the correct order and direction of strokes. The animations were produced automatically using the dynamic features of SVG. The linguistic information were extracted from Jim Breen's dictionaries. At the moment, only about two thousand characters are available on the public web site, before the remaining ones are proof-read.

5. CONCLUSION AND FUTURE WORK

We have described two approaches to make reading and learning Japanese writing easier, by using transliteration and describing *kanji* in terms of their graphical elements. We proposed a Web-based platform that leverages hypertext and vector graphics technology to access and display data that will help foreigners to better read and understand Japanese text on the Web.

There remain however many possible improvements and potential feature additions to this platform, such as: improving the robustness of the transliteration tool for use with real-life web pages; building *learning* material from our data (*e.g.* kanji lessons); improving the appearance of the SVG kanji; improving the user interface. New applications are also planned, including searching for kanji or using the service on small platforms like PDAs and mobile phones.

This work was conducted at the National Institute of Informatics in Tōkō with funding from the Japanese Society for the Promotion of Science.

[5]http://kanjidict.stc.cx/kakasifilt.php
[6]http://www.csse.monash.edu.au/~jwb/kanjidic.html
[7]http://www.google.com/help/features.html#calculator
[8]http://papillon.ex.nii.ac.jp:8998/aaaa/
[9]http://www.w3.org/TR/ruby/