SGML Markup of Japanese Classical Texts: A Case Study

Shoichiro Hara and Hisashi Yasunaga

National Institute of Japanese Literature, 1-16-10, Yutaka-cho, Shinagawa-ku, Tokyo 142, Japan

KEYWORDS: SGML, japanese classical texts, full-text databases

AFFILIATION: National Institute of Japanese Literature

E-MAIL:	hara@nijl.ac.jp
	yasunaga@nijl.ac.jp
FAX NUMBER:	+81-3-3784-8875
PHONE NUMBER:	+81-3-3785-7131

1. Introduction

One of the activities of the National Institute of Japanese Literature (NIJL) has been the designing, building, managing, and maintaining of mainframe-based information systems for Japanese classical literature, made available to researchers in Japan and abroad. At present, we provide three catalogue databases: Catalogue of Microfilms of Manuscripts and Books on Japanese Classical Literature; Catalogue of Manuscripts and Books on Japanese Classical Literature; and Bibliography of Research Papers on Japanese Classical Literature. We have also compiled some full-text databases.

When we began constructing full-text databases, SGML was not yet popular in Japan, and unfortunately there were no SGML applications that could handle the Japanese language. For these reasons, we established our own markup system resembling SGML in its basic conception [1]. We refer to this system as the "KOKIN rules" (from KOKubungaku [Japanese literature] INformation). "Kokin" is also the title of one of the foremost collections of Japanese classical poems.

The KOKIN rules were designed to be easy to understand for researchers of Japanese classical literature and easy for them to use. However, as the markup system is unique and independent, there are no tools available to parse and check KOKIN files. Although SGML was originally a document markup language for publishers, it is now widely accepted as an encoding scheme for transmitting text data irrespective of the platform. Given SGML's international importance and the fact that it has become popular in Japan, we decided that we should convert our text data to SGML markup to enable wider dissemination [2]. We thus began a new project aimed at constructing a full-text database using SGML [3].

Section 2 of this paper describes the KOKIN Rules, and section 3 covers our project to convert KOKIN text to SGML markup. Finally, there is a summary of some problems regarding the markup of classical Japanese texts.

2. Text markup using the KOKIN rules

This section describes the KOKIN rules: the Tag rule, the Flag rule, and the Value-added rule.

2.1 Tag rule

Text is composed of various kinds of logical elements (titles, chapters, etc.). Tags are identifiers that mark up logical elements of text, and the resulting "markup" reflects how a researcher sees and analyzes the text. For example, we see a line as a basic text structure and call it a "logical record". Furthermore, we treat a series of logical records as a "logical record set". Several continuous lines that constitute a poem are an example of the logical record set called "poem". Thus, the KOKIN rules can describe the hierarchical structure of text.

The KOKIN rules were designed to provide guidelines for researchers who want to transcribe primary Japanese classical texts using a computer. There is thus a similarity with the TEI [4]. Our intention was that further text analysis and tag definitions should be performed by researchers themselves. Thus the electronic texts we transcribed define only the basic structure such as titles, pages, and lines. The Tag rule shows how to define the basic structure of a text using the "JapaneseYenMark" followed by a letter of the alphabet (i.e., 'T' for title, 'P' for page, and 'G' to mark the insert position for a graphic element). The following is the syntax of the Tag rule:

```
<Logical Record>    ::= <Tag Begin><Tag><Tag End> |
                        <Tag Begin><Tag><Data>
                        <TagEnd>
<Tag Begin>         ::= "JapaneseYenMark"
<Tag End>           ::= "JapaneseStarMark"
<Tag>               ::= <Tag Symbol> | <Tag Symbol>
                        <Tag Attribute>
<Data>              ::= <Line> | <Original Data>|
                        <Repeating Symbol><Original Data>
<Line>              ::= <Original Data> | <Number>
                        <Original Data>
<Repeating Symbol>  ::= ';'
<Tag Symbol>        ::= .........
<Tag Attribute>     ::= .........
<Original Data>     ::= .........
```

2.2 Flag rule

We believe most Japanese classical texts have a two-dimensional nature – that is, a text is composed of the primary material (titles, body, etc.) and supplementary material (annotations, side notes, etc.). The Flag rule was introduced to define regions of such supplementary material.

The Flag Rule employs the delimiter '/' at the start and end of a word in the primary material to which supplementary material is attached. The supplementary material is enclosed by ‘(‘ and ’)’. The following is the syntax of the Flag rule:

```
<Original Data>       ::= <Flag Begin><Data Element>
                          <Flag End><Supplement>|
                          <Space Flag><Space><Supplement>
<Data Element>        ::= <Paragraph>|<Phrase>|<Word>|
                          <Character>
<Space>               ::= <Space Between Words>|
                          <Space Between Characters>
<Flag Begin>          ::= ’/’
<Flag End>            ::= ’/’
<Space Flag>          ::= ’/’
<Supplement>          ::= <Right Supplement>|<Left
                          Supplement>|<Bi-Supplement>
<Right Supplement>    ::= <Right Supplement Begin>
                          <Supplement Element>
                          <Supplement End>
<Left Supplement>     ::= <Left Supplement Begin>
                          <Supplement Element>
                          <SupplementEnd>
<Bi-Supplement>       ::= <Supplement Begin><Supplement
                          Element> ’|’ <Supplement Element>
                          <Supplement End>
<Supplement Element>  ::= <Single Supplement>|<Double
                          Supplement>
<Single Supplement>   ::= <Supplement>
<Double Supplement>   ::= <Supplement><Supplement
                          Separator><Supplement>
<Right Supplement Begin> ::= ’(’
<Left Supplement Begin> ::= "(|"
<Supplement End>      ::= ’)’
<Supplement Separator> ::= ’#’
<Supplement>          ::= <String>|<String><String
                          Separator><String>
<String Separator>    ::= ’=’
```

2.3 Value-added rule

One of the main purposes of computer-based textual studies is to create a vocabulary index. However, as there are no spaces between words of Japanese text, it is not easy to identify individual words. Thus researchers must perform this task before lexical studies can begin. Unfortunately, different researchers employ different criteria when identifying individual words. Complexity is added by various compound words that appeared over the centuries, resulting in differences from work to work, genre to genre, and period to period. This makes Japanese lexical analysis very difficult.

As mentioned above, we believe that tasks such as these, requiring considerable professional skills, should be performed by the researchers themselves, but our Value-added rule provides assistance for such markup. Using this rule, a researcher can separate a sentence into space-delimited words, adding attributions to any word – such as the phonetic values of Chinese ideographs. These attributions are enclosed by ‘[‘ and ’]’. The following is the syntax of the Value-added rule:

```
<String>              ::= <Value Added Begin><Word>
                          <Value Added End><Value Added>
<Value Added>         ::= <Value Begin><Values><Value End>
<Values>              ::= <Value 1>|<Value 2>|<Supplement
                          Value>|<Value 1><Binding Symbol>
                          <Value 2>
<Value 1>             ::= <Phonetic Representation of a
                          Chinese Ideograph><Attribution 2
                          Begin><Chinese Ideograph>
                          Attribution End>|<Repeating Symbol>
                          <Value 1>
<Value 2>             ::= <Attribution 1 Begin><Variation>
                          <Attribution End><Attribution 2  Begin>
                          ’,’ <Information><Attribution End>|
                          <Repeating Symbol><Element 2>
<Value Supplement>    ::= Not Use
<Variation>           ::= "Part of Speech"  | "Name" | "Location" |
                          "Position"
<Value Added Begin>   ::= ’ ’
<Value Added End>     ::= ’ ’
<Value Begin>         ::= ’(’
<Value End>           ::= ’)’
<Attribution 1 Begin> ::= ’[’
<Attribution 2 Begin> ::= "[,"
<Attribution End>     ::= ’]’
<Binding Symbol>      ::= ’!’
<Repeating Symbol>    ::= ’;’
```

3. Case study

For our case study, we chose to apply SGML markup to an anthology of short stories dating from the Edo period (1600-1868). This text has a complex structure, involving annotations, editorial corrections, phonetic representations, and so on. The original text had been transcribed from a printed edition by a colleague, then marked up using the aforesaid KOKIN rules. This experience had convinced us of the descriptive ability of our markup method, especially for Japanese classical literature. However, the KOKIN rules are essentially abstract, and there are no tools – such as parsers and editors – available for researchers.

Starting in 1993, we applied SGML markup to this same anthology in order to see if SGML was a practical alternative for Japanese classical texts. There were three stages in this case study:

1) creation of a DTD for the anthology;
2) conversion of the original KOKIN text data to SGML; and
3) construction of a full-text database system based on string-search tools.

The tools used were MARK-IT (Sema Software Technology) for data conversion and structure analysis, and OPEN-TEXT (Open Text Co.) for string searching.

3.1 DTD creation

KOKIN-to-SGML data conversion involved replacing the “JapaneseYenMark” and following tag to the START-TAG (<) and END-TAG (>) of SGML. However, text marked up according to the KOKIN rules involves some ambiguities – for example, the open-parenthesis symbol is used, according to the Flag rule, for <Supplement Begin>, but according to the Value-added rule as <Value Begin>. Fortunately, differentiation is easy: examples of the former appear after the ‘/’

of <Flag End>, while the latter appears after the '' of <Value Added End>. Nevertheless, this is context dependent and thus unacceptable under SGML (context-free grammar class).
To resolve these ambiguities, we analyzed the syntax of the KOKIN rules using the E-R model to check precisely the relation between each tag and flag defined in the rules. The DTD we created was checked using a DTD compiler.

3.2 Text data conversion

As the original data had been input and marked up manually, there were many transcription errors and structural inconsistencies that had to be eliminated. Next, we constructed pre-processing algorithms based on the E-R model to convert ambiguous structures to unambiguous ones. Programs were then compiled using these algorithms. Finally, an SGML parser (MARK-IT) was employed to convert pre-processed data to fully SGML-compliant data. MARK-IT checked the input data and added any tags that had been omitted.

3.3 Full-text searching

In contrast to the relatively simple process of data encoding, text searching posed several problems. First, we constructed a full-text database using a relational database system. Though a relational database has standard query tools based on an elegant mathematical model (i.e., SQL, QBE), this type of database imposes fairly strict restrictions on data structure. An object-oriented database is more suitable for complex data structures, but it has neither standard search methods nor any equivalent of SQL.
Nevertheless, as our goal could be described as the ability to "search for a string in a desired region in data," we could construct a full-text database system based on string searching. In the study of Japanese classical literature, searching for strings of interest is a common task. Consequently, we concluded that a full-text database system based on a string-searching system would be effective and practical.
Fast string-searching machines and software are available, and some of these can handle Japanese SGML data. We selected OPEN-TEXT as the basis for our full-text database system.

4. Some problems

Applying SGML to works of Japanese classical literature involves some problems.

4.1 Electronic dictionary

The first problem involves the compound words mentioned above. If we require electronic texts for such intensive study as lexical analysis, spaces must be inserted between words. There are many ways of accomplishing this, such as string matching with dictionaries and the application of artificial intelligence methods. However, none are satisfactory, owing perhaps to the fact that there are no substantial electronic corpora or electronic dictionaries for Japanese classical literature. Of course, the reason why there are no such corpora or dictionaries is that there are relatively few electronic Japanese classical texts available. In other words, we have a chicken-and-egg situation.

4.2 Complex annotations

The second problem involves describing attribute information such as annotations. Many scholars of Japanese classical literature hope that a markup system can encode any original text feature that is important for their research. However, the structure of classical text is very complicated. For example, a string may have two separate annotations; furthermore, the main string and both annotations may be split over two or more lines as follows:

```
      Second Annotation for First Line...
    First Annotation for First Line.......
TEXT LINE on First Line.... Annotated String on First Line....

.......Second Annotation for Second Line
First Annotation for Second Line
...Annotated String on Second Line TEXT LINE on Second Line.....
```

KOKIN Rule describes this structure as follows.

TEXT LINE on First Line..../Annotated String on First Line = Annotated String on Second Line/(First Annotation for First Line = First Annotation for Second Line # Second Annotation for First Line = Second Annotation for Second Line)/ TEXT LINE on Second Line ....

Here, '=' indicates where a string is split over two lines, parenthesis symbols '(' and ')' indicate the starting and ending positions of the annotations, and '#' separates the first and second annotation. The same structure can be described by SGML as follows:

```
TEXT LINE on First Line........................................
  <SuppElement FG="ON">Annotated String on First
        Line</SuppElement>
  <SuppElement FG="ON">Annotated String on Second
        Line</SuppElement>
  <DoubleSupp>
     <FirstSupp>First Annotation for First Line</FirstSupp>
     <FirstSupp>First Annotation for Second Line</FirstSupp>
     <SecondSupp>Second Annotation for First Line
        </SecondSupp>
     <SecondSupp>Second Annotation for Second Line
        </SecondSupp>
  </DoubleSupp>
TEXT LINE on Second Line..........................................
```

Here, the tag <SuppElement> denotes an annotated string region. The attribution "FG" indicates whether or not the annotated string is split over two lines; thus, FG="ON" means the annotated string continues over two lines. <DoubleSupp> after <Supplement> means there are two annotations for the string enclosed by <SuppElement>. <FirstSupp> denotes a first annotation, and <SecondSupp> means a second annotation. Repeated occurrences of <FirstSupp> or <SecondSupp> mean that the annotation is split into two or more lines.

4.3 Non-nested structure

Not infrequently there are missing pages in our original materials. First, we considered treating these missing pages as layout information, using empty tags to indicate them. However, some researchers believe missing pages are essential information. In some cases, they can recover the missing material by referring to other texts. The problem is that if we use ordinary tags to indicate a region of missing pages, these tags will often overlap with another region, as in the following example:

```
.....<Chap>....................................
...........<MissingPage>... Recovered Text .....
....</Chap><Chap>..............................
...........</MissingPage>...</Chap>............
```

Here, if some material is recovered by a researcher, then the <MissingPage> region crosses over into the <Chap> region, thus violating the SGML nesting rule.

4.3 Kanji characters

The most serious problem may be that posed by kanji characters (Chinese ideographs). Some researchers believe that they need over 50 thousand kanji characters to describe all Japanese texts. However, the Japan Industrial Standard (JIS) lists only 12,546 characters, and of these only 6,355 are standard for computer use. Thus, some users, companies and computer centers make their own additional character sets; these characters are called “external characters” since they fall outside the range of standard characters.

Our institute has made about two thousand external characters and more than ten thousand fonts. Printing and displaying these external characters is only possible within the institute, and no elegant solution is yet available to enable data exchange with other sites.

External characters are often used for historical terms, technical terms, and proper nouns such as names. However, there are often alternative Chinese ideographs with the same meaning. Making use of this fact, we employ external characters within the center to publish catalogues that require the correct kanji, but we use alternatives from the standard character set for users on the network who are satisfied as long as they understand the meaning of the original. The same electronic data is used, whether the output is local or on-line, as the external characters are paired with their JIS alternatives, thus:

```
<SelectiveCharacter>
    <![%JIS;[StandardCharacter]]>
        <![%External;[ExtraCharacter]]>
</SelectiveCharacter>
```

Here, “%JIS” must be set to “INCLUDE” and “%External” to “IGNORE” if we want to compile data for network database services, and “%JIS” to “IGNORE” and “%External” to “INCLUDE” for local publishing.

5. Summary

As a test project, we have produced a full-text database and string-search system conforming to SGML. This shows that electronic Japanese classical texts can clearly be constructed on a common SGML model. We are now developing guidelines for transcribing primary Japanese classical texts using SGML. There are still some problems that have to be addressed, but we are hoping to learn from other projects, such as the Wittgenstein Archives at the University of Bergen (WAB), which involves complex formatting issues.

In future, we will determine whether our system can perform satisfactorily under real conditions. In addition, we propose a multimedia database in which texts and images are linked.

References

[1] Hisashi Yasunaga: Data Description Rule and Full-text Database for Japanese Classical Literature, Joint International Conference ALLC-ACH Conference Abstracts, pp.234-239,1992.

[2] Shoichiro Hara and Hisashi Yasunaga: On the Full-text Database of Japanese Classical Literature, Joint International Conference ALLC-ACH Conference Abstracts, pp.61-63, 1993.

[3] Shoichiro Hara and Hisashi Yasunaga: On the Text Based Database Systems for Public Service: Joint International Conference ALLC-ACH Conference Abstracts, pp.43-45, 1995.

[4] L. Burnard et. al.: Guidelines for Electronic Text Encoding and Interchange (TEI P3), ALLC, 1-3, 1994.