CRITAC - A JAPANESE TEXT PROOFREADING SYSTEM

Koichi Takeda
Tetsunosuke Fujisaki
Emiko Suzuki

Japan Science Institute
IBM Japan, Ltd.
5-19 Sanban-cho, Chiyoda-ku,
Tokyo 102, Japan

Abstract

CRITAC (CRITiquing using ACcumulated knowledge) is an experimental expert system for proofreading Japanese text. It detects mistypes, Kana-to-Kanji misconversions, and stylistic errors. This system combines Prolog-coded heuristic knowledge with conventional Japanese text processing techniques which involve heavy computation and access to large language databases.

1. Introduction

Current advances in Japanese text processing are mainly due to the remarkable growth of the word processor market. Machine readable Japanese text can now be easily prepared and distributed. This trend spurred the research and development of further text processing applications such as machine translation and text-to-speech conversion [SAKAS8310] [MIYAG8310]. However, some fundamental text processing procedures are missing for Japanese text. For example, counting the number of words in text is a difficult task since words are not separated by blanks.

Our experimental system CRITAC (CRITiquing using ACcumulated knowledge) tries to overcome Japanese language problems. Proofreading (or critiquing, to some extent) [CHER80] [HEIDJ82] has been chosen as our research domain because it involves many text processing techniques and is one of the most important functions currently required and lacking. In this paper, we introduce CRITAC concepts and facilities including a conceptual representation of Japanese text called "structured text" to handle meaningful objects (e.g., sentences and words) and proofreading using heuristic rules for the structured text. The structured text consists of a set of Prolog [CLOCM81] facts and predicates, each of which represents an object or a class of objects in the text. Because of this high-level representation, human proofreading knowledge can be easily mapped into Prolog rules. Two user-friendly representations of text, called "source" and "KWIC" (Key-Word-In-Context) views, are derived from the structured text. CRITAC provides users with editing and proofreading functions defined over these views.

The notion of structured text, we believe, is not restricted only to the Japanese language. Discussions on our approach for languages other than Japanese will be given in the Conclusion.

2. CRITAC System Overview

In this section we discuss the outline of CRITAC and its underlying concepts. As shown in Figure 1, the heart of CRITAC lies in its architecture. It consists of three major components: a user interface, a preprocessor, and a proofreading knowledge base.

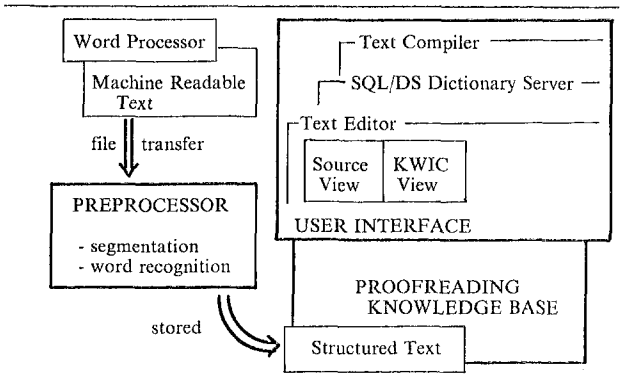

Figure 1. CRITAC Configuration: The preprocessor generates the structured text from given text. The proofreading knowledge base currently consists of about 30 Prolog proofreading rules for the structured text. The user interface handles two external views and facilitates the SQL/DS online dictionary server and text compiler.

User Interface

The user interface is built upon an editor, an SQL/DS online dictionary server, and a text compiler.

The Editor and External Views

The editor provides source and KWIC views as user-friendly external text representations. It facilitates the modification of the text through these views. During the modification, when a user asks the system to apply the proofreading rules, diagnostic messages will appear in the screen with possible errors underlined in the text.

```
＊＊＊ TOP OF FILE ＊＊＊
 1 ワード・プロセッサの普及に伴い。日本語文書を作成することが容易になって
 2 きた。しかしできあがった文書を校正・推敲したりする補助手段を提供するに
 3 は至っていない。仮名漢字変換はうっかりすると非常に漢字の多い文章を作っ
 4 てしまう。ワード・プロセッサにおいて、文書を作成したり印刷したりする機
 5 能は向上しつつあるようであるが、このような校正については、英文では既に
 6 パソコン上で機能するスペル・チェックさえなく、ユーザーは、すべての文を
 7 自分の目で確かめなければならなくなる。その作業は手作業にたより、非常に
 8 手間のかかるものとなる。
 9
10 これは主として、日本語の文書では単語が陽に区切られていないことによる。
11 このため。日本語の性質を考慮した校正方法の研究が重要となる。最近になっ
12 て、文書校正のツールとして、すでに出来上がった文書に対して語用法や文体
13 をチェックするプログラムや、文章の質を評価して、より良い文章に改良する
14 方式などが試作検討されつつある。建石らは、辞書をつかわず、構文解析も行
15 わずに、入力文書と不適当な表現を集めたファイルとを正規表現としてマッチ
16 させることで、ユーザーに注意を促したり、一つの文の長さの最大、平均等に
17 より、文章の読みやすさを測定したりしている。また、石井は入力文をリスト
18 形式に変換し、常用漢字表と朝日新聞の用語調からの知識などをＰｒｏｌｏｇ
```

Figure 2. CRITAC Source View

The source view is just a replica of the original text except that it is not formatted. That is, the original format such as paging and indentations are dropped. Figure 2 shows an example of a source view screen.

The KWIC view displays text in the KWIC format which extracts all the content words as *keywords* from the text and arrange them in their phonetic (pronunciation) order along with their contexts. This is an extremely useful tool for users to find homonym errors [YAMAS8310] caused by a *misconversion*[1] and the lack of conformity (e.g., "center" and "centre" in English) in the text [FUJIM8504]. Since external views are virtual views of the structured text, updates made by the user are checked and reflected in the structured text through the synchronizing mechanism of updates in two different views.

Figure 1 shows a KWIC view. Each line consists of one keyword in the middle, preceding and succeeding contexts are shown on the left and right.

624	一ワードを調べていく校正	規則	が作られている。
625	校正	規則	はＰｒｏｌｏｇのルール
626	あわせて現在２０個の校正	規則	が校正知識ベースとして
627	校正	規則	については特に［５］で
628	作成したり印刷したりする	機能	は向上しつつあるようで
629	英文では既にパソコン上で	機能	するスペル・チェックさ
630	たものを含んだ高度の校正	機能	はいまだに商用かされて
631	よるもので、英文並の校正	機能	を実用かするためには形
632	校正	機能	はこのエディタから呼び
633	・削除・更新が行え、校正	機能	は表示画面上で多面単位
634	ＫＷＩＣエディタの校正	機能	は主として仮名漢字変換
635	ェースの提供するその他の	機能	として、テキスト・コン
636	誤変換・表記のゆれの検出	機能等	は日本語の独自性を扱う
637	接関与して、ミスタイプや	不均一	な語の使用をみつける。
638	もない日本語文書の作成は	近年	急速に機械化されるに至
639	形式を考えており、区切り	記号	や読みの有無はオプショ
640	知識などをＰｒｏｌｏｇで	記述	し、漢字の読み、誤りや
641	を書くときのスタイルか、	句点	から句点、読点から読点
642	きのスタイルか、句点から	句点	、読点から読点までが長
643	終止形で終わっているのに	句点	が打たれていないとき、

Figure 3. CRITAC KWIC View

SQL/DS Dictionary Server

Online access to system dictionaries or an encyclopedia [WEYEB8501] is one of the most user-friendly facilities in an advanced text processing system. CRITAC is connected to a dictionary server implemented on SQL/DS. SQL/DS (Structured Query Language/Data System [IBM8308]) is a relational [CODD7006] database management system. It has been mainly used for business data processing purposes like purchase-order files. The excellent user language of SQL/DS is based on the relational calculus which can be easily incorporated into Prolog [IBM8509]. Furthermore, SQL/DS can support multiple access to tables. For example, a user can access tables in terms of KANJI values or PRONUNCIATION values. This greatly enhances the "associative memory" access of online dictionaries. That is, we can retrieve all the related information by giving some known values.

[1] The Japanese word processors currently have limited the number of keys on the keyboard by using the phonetic character set, Hiragana or Katakana, to enter text. Once the pronunciation of a word or a phrase is given, it will be converted to the most-likely Kanji expression. This process is a Kana-to-Kanji conversion. Because of the large number of homonyms [TANA8310] in the Japanese language, this conversion is liable to generate an unintended Kanji expression. This is referred to as *misconversion*.

We include some "canned" queries to the dictionary. A typical access pattern is searching for homonyms as shown in the section "A Sample CRITAC Session" (Figure 10), which helps users correct misconversions.

Canned queries also include synonyms, antonyms, related words, and upper/lower concept of the given words. The conceptual hierarchy of words is obtained from the following combined SQL queries.

```
SELECT   X.CATEGORY--NUMBER
FROM     SEMANTIC-CLASSIFICATION-TABLE  X
WHERE    X.KANJI = givenWord
```

```
Get the stem number of the returned
category-number.
X.CATEGORY-NUMBER is like "1.2.39".
stemNum = STEM(X.CATEGORY-NUMBER)
```

```
SELECT   X.KANJI,X.YOMI, Y.CATEGORY-NUMBER
FROM     KANJI-PRIMITIVE-WORD-TABLE     X,
         SEMANTIC-CLASSIFICATION-TABLE  Y
WHERE    X.KANJI = Y.KANJI
 AND     Y.CATEGRY-NUMBER LIKE 'stemNum%'
ORDER BY CATEGORY.NUMBER
```

The first query returns the category number, say "1.2.39", from a table called "SEMANTIC-CLASSIFICATION-TABLE" We need a stem number ("1" in this case) of this category. The stem number is then used to find the words whose category number has the same stem number. This is what the second query retrieves. The result is arranged by the category number.

Note that users can make use of not only canned queries but also ad hoc queries. Since the SQL query language and the conceptual scheme of the dictionaries are easy to understand, users can make queries like those above to get ad hoc information. Such a set of customized queries is also stored in SQL/DS. Users can also define views for dictionaries to rename and select columns as well as records.

The Text Compiler

CRITAC also provides a "text compiler" facility. It is analogous to the compilers of programming languages. The user gives text to this text compiler and the compiler can provide the following.

- A source list of text and diagnostic messages with line reference numbers.
- A list of segments and Kanji primitive words with their occurrence counts and pronunciation. A cross-reference list of words and text (KWOC: Key Word Out of Context) and other useful statistics can also be obtained.
- Various formats of the text including KWIC format. Additional information (word boundaries, pronunciation, etc.) may be added to the text. If an application uses a specific format, the text compiler can be made to generate application input in this format.

Preprocessor

The preprocessor decomposes continuously typed Japanese text into a sequence of tokenized primitive words and particles. A basic object of Japanese sentence is a content word followed by zero or more function words, which is called a *segment* or a phrase (see [MIYAG8310], for example, for more details). This preprocess is required because Japanese text has no explicit delimiters (blanks) between words. The preprocessor also gathers such information as pronunciation[2], parts-of-speech, total number of occurrences in the text and base form, etc., associated with each of the words and particles. This information together with the original text is stored as a set of Prolog facts to be used for later processing. We illustrate the steps of the preprocess (see Figure 4).

[Japanese Text Preprocessing]

1. Japanese text is a collection of sentences. Each sentence is just a continuous string of characters.
2. A segmentation algorithm is applied to the sentence. This algorithm contains about 100 heuristic rules each of which specifies the cases where a segment boundary usually appears. The accuracy of this segmentation algorithm is about 97.5%.
3. Content words in the segments are recognized by looking them up in a primitive word dictionary. If a content word is a compound word, it is decomposed into primitive words. Since many Kanji compound words have ambiguities of decomposition, we apply a stochastic estimation algorithm and a Kanji primitive word dictionary with statistics [FUJI8509] to find the most likely decomposition. Our algorithm is obtained from stochastic estimation algorithms in [FORN7303], [BAHLJ8303], and [FUJI8407] with slight modification. The accuracy of our algorithm is about 96.5%.
4. Function words in each segment are identified. The connectivity of these function words is described by an automaton [OKOC8112]. A correct sequence of function words is obtained by observing the transition over this automaton.

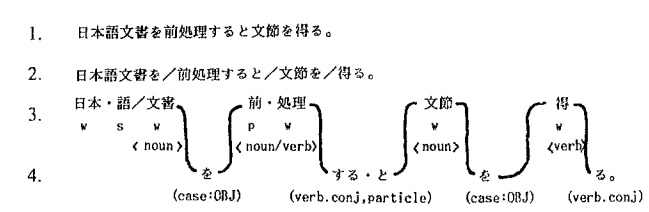

Figure 4. Preprocessing Japanese Text

[2] Roughly speaking, Katakana and Hiragana characters are phonetic and each of them corresponds to one phoneme. Kanji characters are ideographic and there is a many-to-many mapping between the Kanji character set and a set of phoneme sequences.

The preprocess starts with given text at step 1 above. The text is gradually analyzed and decomposed into fragments at the succeeding three steps. Details of the algorithms used here are beyond the scope of this paper.

The above high level objects (segments and words) of Japanese text are conceptually expressed in terms of four types of facts and three types of predicates as shown in Figure 5. We map , <content word>, <function word>, and punctuations into the facts: seg(), head(), tail(), and punc(). Other fragments of text can be defined from those basic facts. This is called *structured text*.

seg(I,J,K,X)	A character string X is the *K-th* segment in the *J-th* sentence of the *I-th* paragraph. I,J and K denote the same indexes below.
head(I,J,K,U,Y,G,L)	U is a *content word* (possibly a Kanji compound word) of the segment X above, with *pronunciation* Y and *part-of-speech* G. L is a list of labels to denote prefixes, primitive words and suffixes in U if U is a Kanji compound word.
tail(I,J,K,V,H)	V is a list of *function words* in the segment X and the part-of-speech of the last function word is H.
punc(I,J,K,D)	D is either a period or a comma of the segment X if any.
sent(I,J,S)	S is *a−sequence−of* segments seg(I,J,K,X).
para(I,P)	P is *a−sequence−of* sentences sent(I,J,S).
text(T)	T is *a−sequence−of* paragraphs para(I,P).

Figure 5. Types of Facts and Predicates for the Structured Text : First four predicates are facts that represent basic objects in Japanese text. The rest of predicates represent derived fragments of the text.

The structured text enables us to generate two external views in the previous subsection (Figure 6). The mapping between the structured text and source view is straightforward. The KWIC view can be denoted by the following symbols.

$$\begin{array}{llllllll} Seg_{1,1} & Seg_{1,2} & \ldots & Seg_{1,k1-1} & Key_1 & Tail_1 & Seg_{1,k1+1} & \ldots\ Seg_{1,W1} \\ Seg_{2,1} & Seg_{2,2} & \ldots & Seg_{2,k2-1} & Key_2 & Tail_2 & Seg_{2,k2+1} & \ldots\ Seg_{2,W2} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \\ Seg_{n,1} & Seg_{n,2} & \ldots & Seg_{n,kn-1} & Key_n & Tail_n & Seg_{n,kn+1} & \ldots\ Seg_{n,Wn} \end{array}$$

The *i-th* row of the KWIC view is a sentence which has W_i segments. $Seg_{i,j}$ is a shorthand notation for a segment X appeared in the *i-th* row of the KWIC view satisfying $seg(p_i, s_i, j, X)$ for some p_i and s_i. Each sentence of the text appears in the KWIC view as many times as the number of its segments because each segment has one keyword to appear in a distinct row in the view. For example, the first sentence appears three times in the KWIC view of Figure 6. Key_x equals the content word of $Seg_{x,kx}$ and T_x is the remaining function words of $Seg_{x,kx}$. If Key_x is a compound word consisting of y primitive words $p_1, p_2, \ldots, p_y$, they are shown in y separate rows instead of a single row with Key_x.

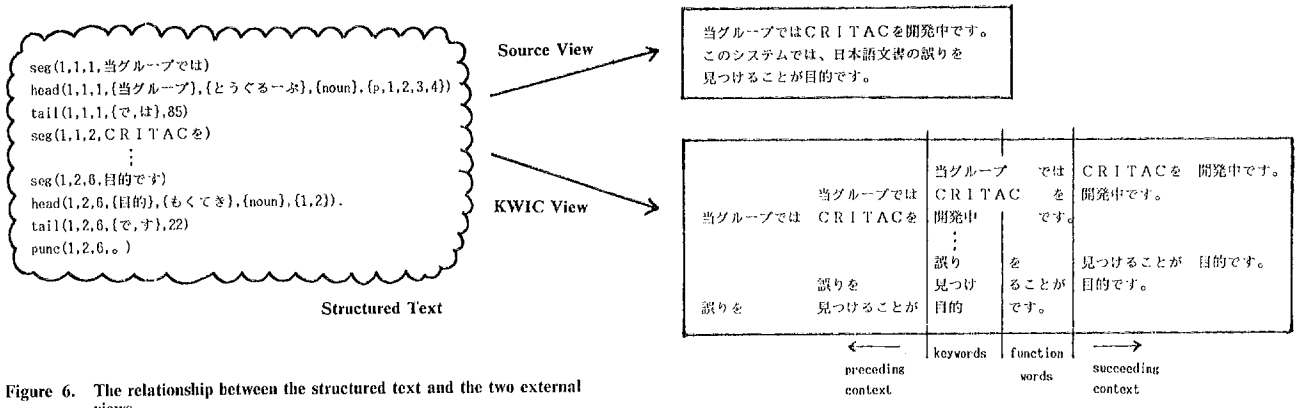

Figure 6. The relationship between the structured text and the two external views

Proofreading Knowledge Base

CRITAC proofreading rules are written in terms of the facts and predicates including those used for the structured text, and built-in predicates [IBM8509]. There are two types of rules: source rules and KWIC rules. By source rules we mean those rules which involve qualification over segments, content words, and function words in the source view of text. The KWIC rules involve the qualification of adjacent keywords as well as their ordering.

Source Rules

One category of source rules aims at finding excessively complicated or ambiguous noun phrases. This is effective in the Japanese language environment because the Japanese grammar allows nouns and some particles to be strung together into a long sequence, and such sequences are used frequently. Basically we have two types of phrases to detect: one is repeated noun modifiers of the same kind and the other is an ambiguous dependency among segments. For example, phrases like

私の母の父の会社の位置は
My Mother's Father's Company's Location is...
(repetition of possesive noun phrase)

等式Ａあるいは Ｂおよび Ｃの適用により
Applying equations *A or B and C*,
(too ambiguous modification)

are detected.

Another category of source rules aims at detecting typographical errors and inconsistent use of words. Some sample proofreading rules are

A rule for detecting incorrect ending of sentences

```
rule('terminative')
        <- tail(I,J,K,T,H)
        & end-of-sentence(E)
        & category('terminative',H)
        & NOT punc(I,J,K,E)
        & warning1('missing punctuation',I,J,K).
```

This rule scans the function words of text. If a segment ends with a function word (the last element of the list T in "tail"), which usually implies the "end-of-sentence", but is not followed by a punctuation mark (period) E, then give a warning to the author.

A rule for detecting an inconsistent use of numeric prefixes - One is Alphabetical prefix and another is Chinese numeric prefix

```
rule('numbers')
        <- head(I1,J1,K1,U1,Y1,G1,L1)
        & number-prefix(K1,L1,P1,W1)
        & head(I2,J2,K2,U2,Y2,G2,L2)
        & number-prefix(K2,L2,P2,W1)
        & char-type(P1,T1)
        & char-type(P2,T2)
        & ne(T1,T2)
        & warning2('numbers',I1,J1,K1,I2,J2,K2).
```

This rule detects the inconsistent usage of Kanji and Roman numeric prefixes for some content word. If some primitive word W1 is preceded by numeric prefixes P1 and P2 denoting the same numbers but not of the same character type, then give a warning to the author. "Number-prefix(K,L,P,W)' succeeds if there is a number prefix P preceding a primitive word W in the compound word K, where the constituent types are listed in L. "Char-type(P,T)" succeeds if a 2-byte character string P consists of only one character type ("Kanji", "Hiragana", "Kanakana", or "Roman") or if P is "mixed".

KWIC Rules

KWIC rules, in contrast with the source rules, refer to keywords of the KWIC view rather than the "segment" or "head" above.

In the KWIC view, as explained in the previous subsection, if Key_1, ..., Key_k is the phonetic ordering, homonyms are arranged adjacently. This greatly reduces the time to detect homonym errors (conversion errors) because the system only has to scan the keywords Key_i once to examine the pronunciation of Key_i and $Key_{i\&1}$, and possibly some other local conditions. For example, we can express possible conversion errors as follows.

```
ruleKWIC('misconversion')
        <- is-ordered('pronunciation')
        & key(I1,U1,Y1,G1)
        & next-key(I1,U2,Y1,G2)
        & ne(U1,U2)
        & pred(I1,P) & pred(I2,P)
        & succ(I1,S) & succ(I2,S)
        & warningKWIC('misconversion',I1).
```

If there are distinct keywords U1 and U2 with the same pronunciation Y1 in the same context P (preceding primitive word) and S (succeeding primitive word), then one of them is possibly a misconversion. "Key(I,U,Y,G)" is Key_I = U with pronunciation Y and lexical category G. "Next-key(I,U,Y,G)" unifies with key(I1,U,G,Y) such that I1 = I + 1.

If Key_1, ..., Key_k are ordered either by their preceding or succeeding word, we can detect some lack of conformity in word usage. For example, if keywords are ordered by their succeeding words, the following case will be detected.

```
... be a       spelling    error  in such cases ...
... can find   style       errors as well as    ...
... to  detect stylistic   errors in the  text  ...
```

```
ruleKWIC('inconsistency')
        <- ordered('successor')
        & key(I1,U1,Y1,G1)
        & next-key(I1,U2,Y1,G2)
        & ne(U1,U2)
        & stem(U1,S) & stem(U2,S)
        & warningKWIC('inconsistency',I1).
```

Here we assume both stem('style','style') and stem('stylistic','style') are successful.

3. A Sample CRITAC Session

This section illustrates a typical CRITAC session consisting of four actions. That is,

1. Detect an error in the source view (Figure 7).
2. Display the explanation of the error (Figure 8).
3. Locate the error candidate in the KWIC view (Figure 9).
4. Get the homonyms of a keyword by invoking the dictionary server (Figure 10).

These actions are implemented as basic functions of the system. Locating a word of interest in different views will efficiently help users judge the system-detected errors to be real errors or not.

```
CRITAC   SOURCE   A1  F 72  TRUNC=72 SIZE=181 LINE=102 COL=1 ALT=0
 97
 98  文書前処理部ではべた書きの文書を（１）文節切り（２）自立部と付属部の分
 99  離（３）漢字複合語の短単位分割（４）付属語の接続検定によって［５］の構
100  造化文書というＰｒｏｌｏｇの節集合に変換する。もとの文書のなかで書式に
101  関する情報は現在ＣＲＩＴＡＣでは扱っていない。書式は文書そのものとは独
102  立に扱え、また出力媒体や文書の用途に応じて容易に指定・変更出来るべきで
                                                    32====================
103  あると考えている。この時点でＣＲＩＴＡＣで扱う前処理語の文書は英文並の
     ======                                    36========
104  区切られた単語の列であるよい。ただし、この文書には書く自立語の読み、自
     15========                                          24====
105  立語や付属語の瀕死などといった情報が埋めこまれている。
106

**CRITAC**|  1 |  2 |  3 |  4 |**|  5 |  6 |  7 |  8 |**|  9 | 10 | 11 | 12 |
  PROOF   |Srce|Next|Quit|Expl|**|SCHG|  ? |Back|Forw|**|  = |Home|Word|KWIC|
====>
```

Figure 7. A Source View with Errors Underlined: Erroneous segments are underlined. The numbers in the underlines are classification code of errors.

```
 CRITAC   SOURCE   A1  F 72  TRUNC=72 SIZE=181 LINE=102 COL=1 ALT=0
+-------------------------------------------------------------------+
| 漢字複合語解析の結果、この自立語の出現確率は                      |
| かなり低いと指摘されています。                                    |
|  ---  誤変換の可能性はありませんか？                              |
+-------------------------------------------------------------------+
100  造化文書というＰｒｏｌｏｇの節集合に変換する。もとの文書のなかで書式に
101  関する情報は現在ＣＲＩＴＡＣでは扱っていない。書式は文書そのものとは独
102  立に扱え、また出力媒体や文書の用途に応じて容易に指定・変更出来るべきで
                                                    32====================
103  あると考えている。この時点でＣＲＩＴＡＣで扱う前処理語の文書は英文並の
     ======                                    36========
104  区切られた単語の列であるよい。ただし、この文書には書く自立語の読み、自
     15========                                          24====
105  立語や付属語の瀕死などといった情報が埋めこまれている。
106

**CRITAC**|  1 |  2 |  3 |  4 |**|  5 |  6 |  7 |  8 |**|  9 | 10 | 11 | 12 |
  PROOF   |Srce|Next|Quit|Expl|**|SCHG|  ? |Back|Forw|**|  = |Home|Word|KWIC|
====>
```

Figure 8. The Source View Explaining One of the Errors: The system explains that the error code 36 says that the underlined segment has a Kanji compound word with very rare combination of primitive words.

```
CRITAC   KWIC     A1  F 116  TRUNC=116 SIZE=1207 LINE=970 COL=1 ALT=0
 960  、より良い文章に改良する      方式          などが試作検討されつつ
 961  本語の性質を考慮した校正      方法          の研究が重要となる。
 962  あとの意味処理につなげる      方法          を検討するつもりである。
 963  検出を行うが、次のような      方法          でこの過程を効率化でき
 964  換や表記のゆれを校正する      方法          を我々はトップ・ダウン
 965  結果は直ちに構造化文書と      他            の外部表現に繁栄される。
 966  ーフェースの提供するその      他            の機能として、テキスト
 967                                本報告        ではこれらの概念とその
 968                                本報告        では日本語文書校正を支
 969  書を校正・推敲したりする      節切          手段を提供するには至っ
 970  時点でＣＲＩＴＡＣで扱う      前処理語      の文書は英文並の区切ら
 971  り、図１に示すように文書      前処理部      、利用者インターフェイ
 972                    文書        前処理部      ではべた書きの文書を（
 973  イプができるため、現行を      見            ながらローマ字漢字変換
 974  、同音意義語の使用状況を      見            たりするのに使われる。
 975  ており、校正メッセージを      見            ながら対話的に文書を変
 976  し呼出し：のような部分を      見            つけ（あるいはＫＷＩＣ
 977  り検出され）もとの文脈を      見            ながら誤りを訂正してい
 978                  （２）の      港            かについても同様である。
 979  々は従来から漢字複合語の      短単位        分割や日本語文の文節切
**CRITAC**|  1 |  2 |  3 |  4 |**|  5 |  6 |  7 |  8 |**|  9 | 10 | 11 | 12 |
  KWIC    |Prf |Yomi|Quit|Expl|**|SChg|  ? |Back|Forw|**|  = |Home|Word|Srce|
====>
```

Figure 9. A KWIC View Switched from the Source View: This KWIC view shows the number of occurrences of a primitive word. The user wonders whether the primitive word caused the previous error 36 of the Kanji compound word.

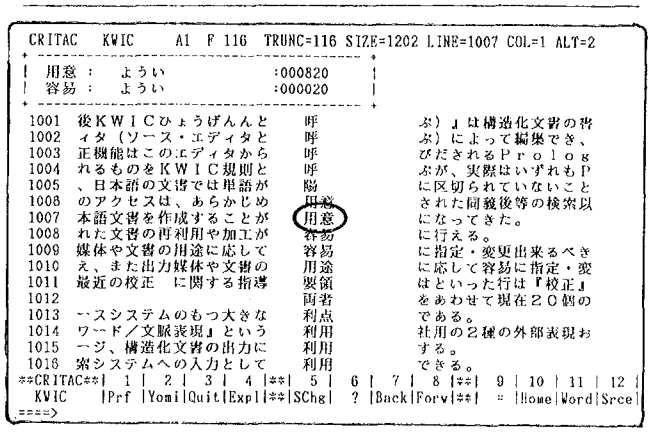

Figure 10. The Invokation of the Dictionary Server: Given a primitive word as a key, the dictionary server displays a list of its homonyms. Then the user might find the correct primitive word in the list.

4. Conclusion

We have introduced a Japanese text proofreading system CRITAC based on the text preprocessing and Prolog-coded knowledge representation. The user interface of CRITAC is developed based on the following concepts.

- support of external views of text for users
- SQL/DS online dictionary server
- text compiler

Although CRITAC is initially designed for the Japanese language, the authors believe it can be applied to other languages as well. For example, the KWIC view can be applied to detect conversion errors in any kind of text prepared by phonetic-to-ideographic conversion or voice-input methods. The notions of structured text and preprocessing not only add textual information to the original text to resolve the overhead of applications but also give a high-level view of the text. The application designer no longer has to code his/her own lexical analyzer that deals with character strings. They are replaced with higher objects such as segment, sentence, or paragraph.

We also believe that CRITAC as a domain-specific (or application-specific) interface to applications (e.g., document retrieval or machine translation) will help users define a set of valid input by heuristic rules in addition to a grammar. Rules of CRITAC can filter out or rewrite some portions of the text and guarantee appropriate text to be given to the applications.

Future work includes

- Evaluation of the system with respect to accuracy, usability and so on.
- A simple grammar checking based on case grammar [FILL68].
- SQL/DS enhancement of dictionary server facilities. SQL/DS can also be used to manage the document [MISE8103] itself.
- Interface to other applications such as a machine translation system or a text-to-voice generation system.

Acknowledgments

We are grateful to Hiroshi Maruyama for building a prototype of the CRITAC knowledge base and for many valuable discussions. We also wish to thank Tetsuro Nishino for implementing a complicated noun analyzer on a CRITAC prototype, M. Arthur Ozeki for his assistance in developing proofreading rules, and Linore Cleveland and Kaoru Hosokawa for their helpful comments on the earlier draft of this paper.

References

[BAHLJ8303] Bahl,L.R., Jelinek,F. and Mercer,R.L.: "A Maximum Likelihood Approach to Continuous Speech Recognition", IEEE trans. on PAMI, Vol.PAMI-5, No.2, pp.179-190, Mar. 1983

[CHER80] Cherry,L.L.: "Writing Tools - The STYLE and DICTION programs", Bell Laboratories Computing Science Technical Report, No.9, 1980

[CLOCM81] Clocksin,W.F. and Mellish,C.S.: *Programming in Prolog*, Springer-Verlag, 1981

[CODD7006] Codd,E.F.: "A Relational Model of Data for Large Shared Data Banks", CACM, Vol.13, No.6, pp.377-387, June 1970

[FILL68] Fillmore,C.: "The Case for Case" in *Universals in Linguistic Theory* (Eds. Bach,E. and Harms,R.T.), Holt, Rinehart and Winston, New York, pp.1-88, 1968

[FORN7303] Forney,G.D.Jr.: "The Viterbi Algorithm", Proc. of the IEEE, Vol.61, No.3, pp.268-278, March 1973

[FUJI8407] Fujisaki,T.: "A Stochastic Approach to Sentence Parsing", Proc. of Coling84, pp.16-19, July 1984

[FUJIM8504] Fujisaki,T. and Morohashi,M. : "Text Processing in Kotodama" (in Japanese) in *Word Processors and Japanese Language Processing* (Eds. Ishida,H. et al.), "bit" special issue, Kyoritsu publishing co., pp.96-107, April, 1985

[FUJI8509] Fujisaki,T.: "Studies on Handling of Ambiguities in Natural Languages" (in Japanese), Doctral dissertation, Tokyo University, Sept. 1985

[HEIDJ82] Heidorn,J, Jensen,K., Miller,L.A., Byrd,R.J. and Chodorow,M.S.: "The EPISTLE text-critiquing system", IBM Systems Journal, Vol.21, No.3, pp.305-326, 1982

[IBM8308] IBM Corp.: *SQL/Data System Concepts and Facilities* (2nd Ed.), GH24-5013, Aug. 1983

[IBM8509] IBM Corp.: *VM/Programming in Logic - Program Description/Operations Manual* Sept. 1985

[MIYAG8310] Miyazaki,M.,Goto,S. and Shirai,S.: "Linguistic Processing in a Japanese-Text-to-Speech-System", Proc. of Intl. Conf. on Text Processing with a Large Character Set, pp.315-320, Oct. 1983

[MISE8103] Misek-Falkoff,L.D.: "Data-Base and Query Systems: New and Simple Ways to Gain Multiple Views of the Patterns in Text", IBM Research Report, RC8769, March 1981

[OKOC8112] Okochi,M.: "Japanese Morphological Rules for Kana-to-Kanji Conversion: Concepts" (in Japanese), TSC Technical Report, N:G318-1560, IBM Japan, Dec. 1981

[SAKAS8407] Sakamoto,Y., Satoh,M. and Ishikawa,T.: "Lexicon Features for Japanese Syntactic Analysis in Mu-Project-JE", Proc. of Coling84, pp.42-47, July 1984

[TANA8310] Tanaka,Y.: "Analysis of Homonyms in Individual Fields", Proc. of Intl. Conf. on Text Processing with a Large Character Set, pp.397-402, Oct. 1983

[YAMAS8310] Yamashita,M., Shiratori,Y. and Obashi,F.: "Kana-to-Kanji Translation Method Using Pattern Characteristics of Kanji Characters", Proc. of Intl. Conf. on Text Processing with a Large Character Set, pp.113-117, Oct. 1983

[WEYEB8501] Weyer,S.A. and Borning,A.H.: "A Prototype Electronic Encyclopedia", ACM Trans. on Office Information Systems, Vol.3, No.1, pp.63-88, Jan. 1985