A NEW INDEXING AND TEXT RANKING METHOD
FOR JAPANESE TEXT DATABASES
USING SIMPLE-WORD COMPOUNDS AS KEYWORDS

Yasushi OGAWA Ayako BESSHO Masajirou IWASAKI
Mina NISHIMURA Masako HIROSE
Research & Development Center, RICOH Co., Ltd.

Abstract

This paper describes a new indexing method for Japanese text databases using the simple keyword string. A compound word is treated as a string of simple words, which are the smallest units in Japanese grammar which still maintain their meanings. As a result, retrieved texts can be ranked, according to the similarity of their meaning and the query, without using a control vocabulary or thesaurus. For automatic indexing, the newly introduced keyword feature, which describes the syntactic and semantic characteristics of a word, enables a precise keyword assignment and simplifies dictionary maintenance.

1 INTRODUCTION

Conventional DBMSs are in wide use for managing well-defined or structured data, however, their use for unstructured data, such as free text, is limited. At the same time, there is an explosion in the volume of text in offices, and companies must now face the problem of dealing with them. In response, we are developing a database management system for Japanese free texts.

Full-text search, as a means for managing texts, is becoming popular. However, there are still substantial problems with these methods: full-text systems generally utilize special hardware, and do not include intelligent search functions such as retrieval of synonyms or text ranking. Our solution is to adopt a keyword-based method, using keywords as indices. We considered two indexing methods[15], controlled vocabulary and uncontrolled vocabulary or free term, and chose the latter. This is due to the fact that a controlled vocabulary requires the creation and maintenance of large dictionaries, and restricts the user to a strictly designated set of words.

In a keyword based system, keywords actually express the contents of the texts and queries, so they should be compared not as literal strings but as meanings. Especially in Japanese where words are easily compounded to express complicated objects, it’s important to include a mechanism for handling compound words in their meanings. In addition, because meanings often are inherently ambiguous and vague, a meaning based matching should be gradationally valued rather than binary. Retrieved texts should also be ranked by grade according to the similarity, or relevance, between the query and the text. As methods for ranking texts based on their content, there are several methods including the vector-space and thesaurus-based methods[15]. The vector space method expresses texts and queries as points in a keyword space. The similarity between a text and a query is defined, for example, by the inner-product of these vectors. However, the method uses select keywords as the axes for the keyword space, and is thus only useful with a controlled vocabulary. Thesaurus based methods enhance a conventional thesaurus by grading relationships among terms, and use these grades to compute the similarity between texts and queries. Fuzzy logic and probabilistic models are used to infer the similarity between texts[7][10][13]. However, all of these methods utilize a thesaurus with only predefined keywords and thus used with a controlled vocabulary.

To match meanings, we propose using a method based on the simple keyword string. A simple word is the smallest unit in Japanese grammar which still maintains its meaning. The most basic system using this would manage each simple word within a word independently. However, since this method ignores the relationships among simple words constituting a compound, it tends to miss necessary texts or to retrieve unnecessary ones. Thus, we treat a compound as a string of simple words and maintain the order of the simple words. The concept of a string of simple words as a keyword is described in Section 2.

The automatic indexing method is described in Section 3. Indexing extracts the important words from a text, and assigns them as keywords for the text. Our method is based on a conventional keyword assignment methods for Japanese text [4][6][11], and modifies it for simple word strings. In conventional methods, nouns or noun phrases are extracted from the target text and unsuitable words are eliminated using a stop word list. However, because creation and maintenance of the list is difficult and a list must be prepared for each application, this method is not practical in a real environment. To solve this, we introduce the concept of the keyword feature, which shows the syntactic and semantic characteristics of a simple word. It is used to evaluate whether a keyword candidate is actually a keyword, so that our method can be expected to attain high precision without a stop word list.

The ranking of retrieved texts is explained in Section 4. Basically, simple words are matched as being either equal or unequal. Compound words are matched using gradations which are computed from the matches of the included simple words. Thus, based on these grades, the similarity of a text is computed according to the proximity of its meaning to that of the query. In this way, the retrieved texts can be ranked without a controlled vocabulary or thesauri.
Since the system has to maintain the order of the simple words within the indexed keywords, a conventional inverted file architecture is not suitable. Instead, we adopt a signature file based method to pre-search the texts to be ranked, while a separate keyword file is used to store the order of the simple words which is used to rank texts. That is, as shown in Figure 1, the signature file is first updated and then automatic indexing is performed during text registration. Text retrieval procedure first pre-searches texts to be ranked using the signature file, and then computes the similarities for those texts using the keywords file. The signature file method adopted here is character-based [5][18], but we modified for Japanese text. This signature file method is explained in Section 5. Evaluation results are given in Section 6.

2 SIMPLE WORD STRING KEYWORD

A simple word is the smallest unit in Japanese text which maintains its meaning, and is equivalent to the β-unit as used by the National Language Research Institute (NLR) [12]. A word contains one or more simple words, and a word with more than two simple words are referred to as a compound word. For example, the Japanese word 物体認識装置 (object recognition device) has three simple words: 物体 (object), 認識 (recognition), 装置 (device). In our system, it is treated as a string of simple words, expressed as 物体/認識/装置.

The component simple words can be easily dealt with independently: during text registration, each simple word is assigned as an independent keyword. During text retrieval, the user’s request can also be segmented into simple words and the final result is generated from retrieved texts of component words (intermediate results). There are two methods to generate the final result; one is to AND the intermediate results, and the other is, to OR the results. In the case of AND, the final result is the set of texts which contain all of the component words in the query. Thus texts which are assigned only a part of query simple words cannot be retrieved. In the case of OR, the final result is the set of texts which contain at least one of the components in the query. Therefore, the result usually contains so many undesirable texts. However, since a simple word string holds its containing simple words, a retrieval based on this string can exclude the falsely matched texts from the above naive method. In addition, retrieved texts can be graded as we explain in Section 4.

Another important advantage in using simple words as the unit is that, during automatic indexing as described in the next section, a lexical parser is used to divide text into words. This requires only a relatively small dictionary containing only simple words. Based on a word count of newspaper articles by the NLR [12], the simple word count was approximately 47,000 while the compound word counting was nearly 101,000. Furthermore, the most frequent 10,000 (compound/simple) words only accounted for 79.5% of the compound words, whereas there was 92.6% coverage by the simple words. This makes simple words very favorable for system applications.

3 AUTOMATIC INDEXING METHOD

To simplify the text registration procedure for the users, the system must provide automatic indexing. Conventionally, free term indexing methods for Japanese texts parse texts lexically (and syntactically) and extract nouns or nouns phrase from the target text [4][6][11]. We have modified this method to be applicable for simple word strings. The overall flow of the indexing is as follows:

1. Lexical and Syntactic Parsing:
   Since Japanese text contains no obvious space between (simple) words, the text must be lexically and syntactically parsed to identify simple words. Our system utilizes the Japanese parser developed at our Research Center [3]. A part of speech and keyword features, which are explained later, are also assigned to each simple word.

2. Keyword Candidate Extraction:
   Based on the part of speech, simple words are extracted from the parsed text as keyword candidates.

3. Keyword Candidate Evaluation:
   Unnecessary simple words are deleted from the candidate, resulting in the actual keywords.

3.1 Keyword Candidate Extraction

Simple words are selected from the parsed text based on their parts of speech. A keyword candidate is formed by successive simple words whose parts of speech are one of the followings:

- nouns

Japanese includes four kinds of nouns: common nouns, proper nouns, verbal nouns and copular nouns [9]. The latter two are unique to Japanese. A verbal noun is...
a noun which constitutes the stem of a compound verb when followed by する (sura) or one of its conjugative forms. A copular noun (adjectival noun) is a noun which constitutes the stem of a compound adjective when followed by なる (na) or one of its conjugative forms.

- prefixes
- suffixes
- the particle の (no; "of" or "s")
- numerals
- numeratives
- unregistered

A string of characters which cannot be identified during parsing is treated as unregistered.

3.2 Keyword Candidate Evaluation

Keyword candidates may include unsuitable simple words; whether a word is a keyword or not is determined not only by its part of speech but also by its syntactic and semantic characteristics. However, candidates are extracted only based on simple word's part of speech. Therefore, we have to evaluate keyword candidates and eliminate unnecessary simple words from them.

3.2.1 Keyword Features

To address the above problem, we introduce keyword features which describe the characteristics of each word more precisely than part of speech. They are used in the mark-up rules of the elimination process and also in text ranking.

Currently, we use the following four features:

1. Modifier:
   This feature is attached to nouns and prefixes. It is assigned to nouns where the word seldom appears as an isolated noun and is commonly used as the modifier within a compound noun. For example, 国際 (international) is a common noun with the modifier feature. This word is usually found in compound words such as 国際関係 (international relationship) and 国際会議 (international conference).

   This feature is also assigned to prefixes which commonly, strongly modify the succeeding words and serve to discriminate the meaning of the compound word.

2. Compound Noun Base (CNB):
   This feature is assigned to nouns with rather broad meanings which commonly serve as the base or stem of compound words. For example, a common noun システム (system) has this feature. It is usually found in compound words such as データベース システム (database system), 画像/処理 システム (image processing system).

3. Less-Discriminating Place name (LDP):
   This feature is attached to copular nouns or place name which discriminate to a lesser degree. This feature is attached to Japanese prefectures, large cities, etc.

4. Electronics Field (ELF):
   This feature is assigned to numeratives found in documents particular to electronics field. This includes terms such as ビット (bit), ペイト (byte).

   A keyword feature can be assigned to a simple word, and is managed in a separate file from the parsing dictionary. The feature makes it much easier to modify or tune the automatic indexing to specific applications since the indexing criteria can be easily modified by changing the indexing rules.

3.2.2 Elimination Algorithm

Unnecessary simple words in a keyword candidate are eliminated based on their keyword feature as well as parts of speech. Relationships among successive simple words are also used to evaluate a keyword candidate[1].

We, first, classify simple words into three classes:

_TYPE1 : the particle の, and prefixes without any features,
_TYPE2 : common and proper nouns with modifier, CNB or LDP feature, verbal and copular nouns, prefixes with modifier feature, suffixes, and numeratives without any features,
_TYPE3 : common and proper nouns without any features, numeratives with ELF feature, and unregistered.

Numeral parts are combined into one unit and classified according to the following simple word: When a numeral part is followed by a ELF numerative, the part is categorized as TYPE3. If the following word is a numerative without ELF or a non-numerative, the part is recognized as TYPE1.

Each keyword candidate is processed as follows:

1. Mark-up simple words with TYPE1, TYPE2 or TYPE3,
2. Eliminate TYPE1 simple words,
3. Eliminate TYPE2 simple words:

   A TYPE2 word is removed only when the length of the candidate string is 1 after all TYPE1 words are deleted.

Note that a TYPE3 word is never eliminated.

3.3 Example

Consider the following sample text: リコーの研究開発所は超音波センサを使った3次元物体識別装置を開発した。

(Ricoh's Research and Development Center has developed a 3-dimensional object recognition device using ultra-sonic sensors.)

The parsing result and its word-by-word translation is shown in Table 1. The “mark” column shows the intermediate results:
Table 1: Result from Parsing and Mark

<table>
<thead>
<tr>
<th>simple word</th>
<th>part of speech</th>
<th>feature</th>
<th>mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>リコーハ (Ricoh)</td>
<td>proper noun</td>
<td>CNB</td>
<td>3</td>
</tr>
<tr>
<td>の (s)</td>
<td>particle</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>研究 (Research)</td>
<td>verbal noun</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>開発 (Development)</td>
<td>verbal noun</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>所 (Center)</td>
<td>particle</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>は</td>
<td>prefix</td>
<td>modifier</td>
<td>2</td>
</tr>
<tr>
<td>音波 (sonic)</td>
<td>common noun</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>センサ (sensors)</td>
<td>common noun</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>を使う (using)</td>
<td>verb</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>た (3)</td>
<td>aux. verb</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>次元 (dimensional)</td>
<td>numeral</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>物体 (object)</td>
<td>common noun</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>識別 (recognition)</td>
<td>verbal noun</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>装置 (device)</td>
<td>common noun</td>
<td>CNB</td>
<td>2</td>
</tr>
<tr>
<td>を開発 (developed)</td>
<td>particle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>し (has)</td>
<td>aux. verb</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A black entry indicates the corresponding word is not extracted as keyword candidate, and "1", "2" and "3" stand for marks TYPE1, TYPE2 and TYPE3.

For instance, in the candidate string / 3 / (Research/Development/Center), where a numeral 3 is followed by a numerative 次元 which has no feature, both 3 and 次元 are marked as TYPE1 and eliminated from the final keyword. Even 識別 and 装置 are marked as TYPE2, the string length is 3 even after deleting 3 and 次元, and these words are included in the result. On the other hand, the candidate 開発 (developed), marked as TYPE2, is eliminated because its length is 1. The final result contains the three keywords: (1) / リコーハ/研究/開発/所/Ricoh research development center, (2) / 音波/センサ/ (ultra sonic sensor) and (3) / 物体/識別/装置/ (object recognition device).

4 TEXT RANKING METHOD

As explained in Section 1, since it is difficult to simply evaluate a text as either matching the query or not, our system grades and ranks retrieved texts. Our ranking method differs, however, from those using the vector space model or thesaurus [15]. The method does not require a controlled vocabulary or a large thesaurus, and relevance values for texts can be determined from simple rules.

The ranking is performed as follows:

1. Extracts simple word strings from the query:

The users can input any noun phrase. The system parses the input and extracts a simple word string keyword using the method described in Section 3.

2. Assigns a weight to each simple word in the query:

The weight shows the relative importance of each simple word within the query word string.

3. Compute text relevance value for all pre-searched texts:

The text relevance value (TRV) grades the similarity between a query and a text and is calculated from the keyword relevance values (KRVs) for keywords in the text. KRV is a grade of similarity of an indexed keyword to the query keyword.

To compute the TRV, indexed keywords are read from the keyword file. To reduce the amount of I/O operations to speed up the retrieval process, our system introduce a pre-search, finding texts which are likely to have non-zero TRV. The pre-search method is explained later.

4. Sort texts in descending order of TRVs.

After computing TRVs for all pre-searched texts, texts are sorted in order of TRVs.

In the rest of this section, we detail the methods for weight assignment, KRV and TRV computations.

4.1 Weight Assignment

At this stage, the user's input has been parsed and recognized as a simple word string. The importance of each simple word in the query string differs from others because each word has distinct syntactic and semantic meanings. Thus, we assign to each simple word a weight to represent its relative importance.

We classified simple words into three categories on their parts of speech and features. We introduce two pre-determined values, initial value \( \eta \) and increasing value \( \delta \), then weights are assigned in the following order:

CLASS1 : Prefixes and suffixes

For prefixes, the weight is set to \( \eta \) if the prefix has the modifier feature, or to 0, otherwise. All suffixes are weighted by \( \eta \).

CLASS2 : Common and proper nouns with features, verbal nouns, copular nouns, numerals and numeratives

We introduce a variable called point, initially set to \( \eta \). The simple word string is inspected from its tail to its head. If a simple word of this class is found, its weight is set to point, and point is increased by \( \delta \) for the next simple word.

CLASS3 : Common and proper nouns without features

We again use a variable point. First, point is set to the sum of assigned weights of CLASS1 and CLASS2, and increased by \( \delta \). Then, words belonging to this class are processed from the tail to the head. If one finds a word of the class, point is assigned to it and point = 2 x point + \( \delta \).

4.2 Keyword Relevance Value Computation

As explained before, keyword relevance values (KRV) grades the similarity between two keyword strings. A larger KRV
means that two keywords are more similar to each other. It is used to compute text relevance value (TRV).

In the following, the query keyword \( Q \) consisting of \( m \) simple words will be expressed as \( q_1 \cdots q_m \), and the indexed keyword \( K \) of \( n \) words, as \( k_1 \cdots k_n \). Then, we introduce a following values, \( krv_Q(K) \), to express the similarities of \( K \) to \( Q \):

\[
krv_Q(K) = \prod_{i=1}^{n} w(k_i) \times \alpha^v(Q,K)
\]

where \( w(q) \) is a weight of the simple word \( q \), and yields 1 when \( q \) is not included in the query. \( \alpha \) is the pre-determined adjacent value. \( v(Q,K) \) gives the number of adjacencies of query's simple words which coincide with keyword's ones. For example, \( v(Q,K) = 1 \) when \( Q = /A/B/C/ \) and \( K = /C/A/B/D/ \), because the adjacency /A/B/ is found both \( Q \) and \( K \).

In the formula above, the first term, word point, reflects the importance of words which are common in the query and indexed keywords. The second term, adjacent point, is to distinguish the order of two simple words which are found both in the query and indexed keywords. That is, because the order of simple words is ignored in computation of the first term, there is no way to distinguish two compound words such as /A/B/ and /B/A/.

Assume that we define KRV by the above value, comparing KRVs for different queries has no meaning because weight assignment results vary according to the queries. This means such KRVs are not useful as absolute measurements of the similarity of each text vis-a-vis the queries. Thus, the final KRV is defined as the above value divided by a query point, \( krv_Q(Q) \):

\[
KRV_Q(K) = krv_Q(K)/krv_Q(Q)
\]

### 4.3 Text Relevance Value Computation

A text relevance value (TRV) grades the similarity between a query and a text, and is computed from the KRVs for the keywords indexed to the text. TRV is defined as the maximum of the KRVs:

\[
TRV_Q(T) = \max_{K \in T} KRV_Q(K)
\]

### 4.4 Example

We illustrate how the TRV is computed. First, we set parameters used in the computation: the initial value \( \eta = 2 \), the increasing value \( \delta = 1 \), the adjacent value \( \alpha = 2 \).

We choose \(/\) 物体/\)/ \( /\) 言語 / \( /\) 装置 / \( /\) 認識/\)/ \( /\) 認識(認識) / \( /\) 装置(認識) / \( /\) 裝置(認識) (object recognition device) as an example query keyword, and the result is shown in Table 2. 認識 and 裝置, belonging to CLASS2, are processed first; since 認識 precedes 裝置, \( w(\)認識) becomes \( w(\)装置) + 1 = 3. 物体 is the only word of CLASS3, its weight is set to \( (3+2)+1 = 6 \) (the sum of weights of 認識 and 裝置, increased by 1). The query point for this keyword is given by \( (6 \times 3 \times 2) \times 2^2 = 72 \).

|| simple word | part of speech | feature | weight |
|-------------|--------------|---------|--------|
| 物体 (object) | common noun | 6      |
| 認識 (recognition) | verbal noun | 3      |
| 装置 (device) | common noun | 2      |

Then the TRV is computed. We assume a text which has two indexed keywords: /\)物体/\)/ \( /\)表面/\)/ \( /\)表面(表面) / \( /\)面/\)/ \( /\)認識/\)/ \( /\)認識/\)/ \( /\)装置/\)/ \( /\)装置/\)/ \( /\)装置/\)/ \( /\)装置(認識) (object surface) and /\)文字/\)/ \( /\)認識/\)/ \( /\)認識(認識) / \( /\)認識(認識) (character recognition device). Firstly, the KRV for 物体表面 is computed as follows: Only a simple word 物体 is in common between the query and this word, so the word point is \( w(\)物体) = 6 and the adjacent point is 1. Thus, the KRV becomes \( 6/72 = 0.083 \). Next, for 文字認識装置, because words 認識 and 裝置 are matched the word point is \( w(\)認識) \times w(\)装置) = 6. Since a word adjacency between 認識 and 裝置 is also matched, the adjacent point is 2. Thus, the KRV is \( (6 \times 2)/72 = 0.167 \). As a result, the TRV for the text becomes 0.167.

### 5 PRE-SEARCH METHOD USING CHARACTER-BASED SIGNATURE METHOD

Usually an inverted file is used for text management using keywords. However, the naive inverted file method cannot be used here because the order of the simple words in a keyword must be maintained. However, some access method is still needed to achieve high performance. We have adopted another indexing method based on the signature method [2].

#### 5.1 Character-based Superimposed Signature Method and Its Problem

Usually, a text signature is computed from words in the text [2]. Since there is no obvious separation between the words in Japanese text, we adopt the signature method based on characters [5][18]. More precisely, the method is as follows:

- Each character is treated as one word, that is, a character is the unit of computing signature,
- The bit width of a signature is set to the size of alphabet,
- A hash function is selected so that it yields a signature where only one bit corresponding to a character is set,
- A text signature is a superimposed or ORed result of all characters' signatures of a text.

We call this method the Character-based Superimposed Signature (CSS) Method.

One of the drawbacks of signature methods, including CSS, is false drop, which retrieves some texts which do not actually qualify the query condition. This problem is usually solved by expanding the signature size. For CSS, it is solved by generating the signature from two adjacent characters, as is done in the Map Array Signature (MAS) [18]. However,
Japanese alphabet is so large (about 7,000) that the resulting signature size becomes huge \(7,000^2 = 49 \times 10^6\). Thus we cannot directly use the MAS method for Japanese texts.

### 5.2 Extended CSS Method

To lower false drop while keeping the signature size small, we propose an extended CSS method. In this method, we use two signatures:

- **Adjacent character signature (ACS):**
  
  A newly introduced signature is made using an adjacent code. An adjacent code is the concatenation of the lower bytes of adjacent characters. The signature is made in the same way to generate SCS.

- **Single character signature (SCS):**
  
  This is the same as the conventional character signature. Since the ACS signature uses only the lower byte, false drops increase compared to MAS. Thus we use SCS at the same time.

Figure 2 illustrates the configuration of these signatures. The total size of the signatures becomes \(7,000 + 27^7 = 23 \times 10^3\), and is much smaller than MAS. In addition, these signature files are compressed using the Exp-Golomb algorithm.

When a text is registered, the signatures for the text are generated as described below, and added to the corresponding files. In the following, the text is represented by \(T = t_1, \ldots, t_t\), where \(t\) stands for a character. For the SCS, \(S_{\text{single}}(c, T)\), a bit corresponding to character \(c\) for \(T\), is set by:

\[
S_{\text{single}}(c, T) = \begin{cases} 
1 & \exists t_j, c = t_j \\
0 & \text{otherwise}
\end{cases}
\]

Let \(F(c_1, c_2)\) represent a function which concatenates lower bytes of characters \(c_1\) and \(c_2\). For the ACS, \(S_{\text{adjacent}}(d, T)\), a bit corresponding to adjacent code \(d\) for \(T\), is set by:

\[
S_{\text{adjacent}}(d, T) = \begin{cases} 
1 & \exists t_j, d = F(t_j, t_{j+1}) \\
0 & \text{otherwise}
\end{cases}
\]

During text retrieval, the result is obtained in the same way as the conventional method. Let \(Q = q_1, \ldots, q_m\) be the query

\[
\{ \bigcap_{i=1}^m L_{\text{single}}(q_i) \} \bigcap \{ \bigcap_{j=1}^{m-1} L_{\text{adjacent}}(F(q_j, q_{j+1})) \}
\]

### 6 EVALUATION

We have implemented the described automatic indexing method, text ranking method and pre-search method on UNIX workstations. Then, we have evaluated these methods.

We prepared 20,000 news articles, and selected 200 articles (50 electronics related articles and 150 others) randomly for detailed evaluation. The electronics articles were separated from others because some keyword features are assigned simple words which are found mainly in this field. The average length of the articles is 650 Japanese characters (that is, 1.3 K bytes).

#### 6.1 Effectiveness Measurement

Usually the effectiveness of automatic indexing or information retrieval is measured by the recall and precision rates. While the recall shows the ability of the method to present all relevant keywords or texts, the precision, the ability to present only the relevant keywords or texts. The recall \(R\) and the precision \(P\) are defined by:

\[
R(\%) = \frac{\#(\text{Relevant selected keywords/documents})}{\#(\text{All relevant keywords/documents})} \times 100
\]

\[
P(\%) = \frac{\#(\text{Relevant selected keywords/documents})}{\#(\text{Selected keywords/documents})} \times 100
\]

where \(\#(X)\) is the number of \(X\).

#### 6.1.1 Automatic Indexing

We compute recalls and precisions individually for each of the 200 articles, then take the average. Extracted keywords are manually evaluated as either relevant or not. Because the system assigns a keyword as a simple word string, we judge automatically indexed keywords in two ways: a partial match indicates that an extracted keyword is judged as relevant when at least one component simple word in the keyword is found in the manually assigned keyword, and a complete match indicates the word strings are completely the same as the corresponding keyword manually assigned.

To evaluate the effectiveness of keyword features (KFs), we compared \(R/P\) for indexing results using KF and not. The results are shown in Table 3. Comparing both methods, although \(Rs\) are slightly decreased \((96.8 - 94.5)/96.8 = 2.3\%\) for average in complete match), \(Ps\) are greatly improved.

---

2 Actually, we only use the lower seven bits, because the MSB of the lower byte is not used in EUC (Extended Unix Code).
Table 3: Automatic Indexing Evaluation Results

<table>
<thead>
<tr>
<th></th>
<th>Electronics</th>
<th>Others</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Par. (no KF)</td>
<td>97.6</td>
<td>48.1</td>
<td>96.5</td>
</tr>
<tr>
<td></td>
<td>95.3</td>
<td>58.2</td>
<td>94.2</td>
</tr>
<tr>
<td>Com. (no KF)</td>
<td>68.8</td>
<td>17.4</td>
<td>67.2</td>
</tr>
<tr>
<td></td>
<td>64.3</td>
<td>26.5</td>
<td>60.8</td>
</tr>
</tbody>
</table>

There is linear relationship between the text length and response time. Table 5 gives registration speed calculated using the relationship.

Table 4: Text Ranking Evaluation Results

<table>
<thead>
<tr>
<th></th>
<th>Electronics</th>
<th>Others</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple AND</td>
<td>57.1</td>
<td>82.5</td>
<td>64.8</td>
</tr>
<tr>
<td>Simple OR</td>
<td>100</td>
<td>60.0</td>
<td>70.0</td>
</tr>
<tr>
<td>Top 10</td>
<td>82.4</td>
<td>86.0</td>
<td>84.6</td>
</tr>
</tbody>
</table>

6.2.1 Text Registration

Text registration consists of an update of the signature files and the keyword assignment. We measure registration time, and there is linear relationship between the text length and response time. Table 5 gives registration speed calculated using the relationship.

Table 5: Registration Speed

<table>
<thead>
<tr>
<th></th>
<th>Speed (chr/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature File Update</td>
<td>135.1</td>
</tr>
<tr>
<td>Keyword Assignment</td>
<td>140.9</td>
</tr>
<tr>
<td>Total</td>
<td>69.0</td>
</tr>
</tbody>
</table>

Table 6: Pre-search Speed

<table>
<thead>
<tr>
<th>Length of Character String</th>
<th>Speed (text/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cold start</td>
<td>69/100</td>
</tr>
<tr>
<td>warm start</td>
<td>102/00000</td>
</tr>
<tr>
<td>1</td>
<td>15/800</td>
</tr>
<tr>
<td>2</td>
<td>89/700</td>
</tr>
<tr>
<td>3</td>
<td>93/700</td>
</tr>
<tr>
<td>4</td>
<td>28/300</td>
</tr>
<tr>
<td>5</td>
<td>32/000</td>
</tr>
</tbody>
</table>

6.2.2 Text Retrieval

We measure response time for a cold start, which runs the sample query with an empty cache, and a warm start, which runs with the cache filled with the necessary data.

Text retrieval also consists of two phases, pre-search and ranking. First, the measurement of pre-search is summarized in Table 6. We can see that the speed is fast enough, especially with a warm start. In a real environment, because we might expect a warm cache, the pre-search method is suitable for our system.

Then we show the total performance including text ranking. As a result of measuring the response time for 200 queries, we find a relationship between the number of pre-searched texts and the response time. Using this relationship, we compute the retrieval performance (Table 7). As the pre-searched texts increase, the speed decreases, because the more indexed keywords have to be accessed to compute TRV for the pre-searched texts.

7 CONCLUSION

We proposed the new indexing and text ranking methods for Japanese text database using the simple keyword string. Conventional keyword-based text management systems require a controlled vocabulary or a thesaurus for retrieving texts in their meanings. However, since our method uses simple words, the smallest units in Japanese grammar which still maintain their meanings, as the unit of text management, the method can retrieve texts in order of their similarity to the query. In order to describe the syntactic and semantic nature of each simple word more specifically than part of speech, we introduced the keyword feature. It is used to evaluate keyword candidates so

---

Footnote: 4We set ranking parameters as follows: $\eta = 2$, $\delta = 1$, and $\alpha = 2$. 

Footnote: 5Because, since the length of one Japanese character is 2 Byte, the speed should be doubled for byte/sec.
that our keyword extraction method can eliminate unsuitable simple words in the candidates without a stop word list. It’s also utilized in computing relevance value of two keywords during text ranking.

The method was confirmed to be effective by measurement results of recall and precision rates. In automatic indexing, the keyword evaluation process based on the keyword features effectively eliminated unsuitable simple words in keyword candidate; compared with keyword candidates, the final keywords attained much higher precision while recalls were slightly decreased. In addition, the nearly perfect recall rate means the method extracted almost all of the necessary keywords from the text. The evaluation results confirmed our retrieval method ranked relevant texts in higher position than irrelevant ones. Precisions were greatly improved for Top 10 texts from all pre-searched texts.

To speed up retrieval, we utilized the pre-search procedure using the character-based signature (CSS) method. Because our method introduces the adjacent character signature, the method achieves lower false drop rate than the conventional CSS, and keeps the signature size much smaller than the MAS.

References


