GENERATING SUPPLEMENTARY INDEX RECORDS USING MORPHOLOGICAL ANALYSIS FOR HIGH-SPEED PARTIAL MATCHING

Masahiro Oku

NTT Affiliated Business Headquarters
20-2 Nishi-shinjuku 3-Chome Shinjuku-ku, Tokyo 163-1419 Japan
E-mail: oku@nbd.hqs.ntt.co.jp

and

Ryousuke Noda and Ryoji Nagai

NTT Information and Communication Systems Laboratories
1-1, Hikari-no-oka, Yokosuka-shi, Kanagawa 239-0847 Japan
E-mail: {noda, nagai}@isl.ntt.co.jp

ABSTRACT

The telephone directory retrieval task requires high-speed partial matching, and many processes run concurrently so there is a need to reduce resource requirements such as CPU power. To realize the high-speed partial matching that satisfies these requirements, this paper proposes a method for generating supplementary index records. This method enables us to retrieve all candidates including all words within strings input by users from a DB without applying morphological analysis to the input strings on the fly. Moreover, this paper describes the results of evaluation experiments. The experimental results show that the proposed method requires about 60% less CPU power and 80% less logical disk I/O than the simple partial matching, and that the proposed method takes less than about 5 seconds when fewer than 100 records are retrieved. Namely, the experimental results show that the proposed method satisfies the requirements mentioned above quite well. This means that the proposed method is practical when applied to the telephone directory retrieval task.

1. INTRODUCTION

Higashida [4] has been developing a fully automated Japanese telephone directory assistance system using touch-tone telephones as input devices [3] [4]. In the written Japanese language, there are no spaces between words. Therefore, Japanese morphological analysis is necessary to recognize word boundaries for input strings. However, since morphological analysis requires high-cost processing, it is not suitable for the telephone directory retrieval task in which processing speed is very important.

Let' s consider that a company name is input as the retrieving keyword by a telephone directory user. If the user knows and inputs the complete company name, the system can exactly match the keywords input against the index records in the DB. However, many users know only part of the company name. In this case, a partial string of the company name is input as the retrieving keyword. Namely, it is unusual for there to be an exact match between retrieving keywords and index records and partial matching is the fundamental operation for the telephone directory retrieval task.

Consequently, the requirements for telephone directory retrieval task are as follows:

- **High-speed partial matching:**
  In the telephone directory retrieval task, it is necessary to obtain retrieval results from incomplete input strings. In order to satisfy this requirement, we have to perform partial matching. In the telephone directory retrieval task, the number of records in the subject DB is of the order of several million. Therefore, we have to develop an extremely high-speed partial matching method.

- **Supporting many processes concurrently by reducing resource requirements:**
  Since many tasks must be handled at the same time in the telephone directory retrieval task, the system must invoke many processes concurrently. Therefore, it is indispensable to reduce resource requirements such as CPU power, memory, and disk I/O.

To satisfy the above requirements, this paper proposes a method for generating supplementary index records using Japanese morphological analysis for high-speed partial matching. The supplementary index records are managed by a commercial DBMS. This allows us to realize high-speed partial matching since the DBMS can handle the supplementary index records by traversing the index tree efficiently.

2. DEFINITION OF KEY TERMS

Key terms used in this paper are defined as follows:

- **Retrieving keyword:**
  " retrieving keyword" is the matching string input by a user for retrieving one or entries from a subject DB.

- **Index file and index record:**
  " index file" is a set of index records. " index record" consists of at least two fields, i.e. a name field and a pointer field. The name field is a string to be matched against a retrieving keyword. The pointer field points to the corresponding information for that record in the subject DB. In the following, we simply refer to the name field of an index record as the " index record" .

- **Partial matching:**
  When the whole retrieving keyword is included in a index record, the retrieving keyword succeeds in " partial matching" for the index record. For example, as the retrieving keyword = " *denwa* (telephone)" is included in index records " nippon-*denwa* (Japan telephone)" , " *denwa*-gaisha (telephone company)" and " denshin-*denwa*-gaisha (telegraph telephone company)" , " *denwa* (telephone)" partially matches these index records [1]. Moreover, retrieving keyword = " *kyouto*" partially matches both index records " *kyouto*-fu" and " tou*kyou-to*" . In the latter case, " *kyouto*" partially matches the combination of a part of two words since " toukyou-to" consists of two words, " toukyou" and " to" . Such cases degrade the precision rate of partial matching.

- **Partial matching from the beginning:**
  The case that a retrieving keyword partially matches an index record from the beginning character is called " partial matching from the beginning" . For example, retrieving keyword " *denwa* (telephone)" succeeds in " partial matching from the beginning" for the index record " *denwa*-gaisha (telephone company)" .

- **Partial matching from the middle:**
  All partial matching cases other than from the beginning are called " partial matching from the middle" . For example, retrieving keyword " *denwa* (telephone)" performs " partial matching from the middle" for the index records " nippon-*denwa* (Japan telephone)" and " denshin-*denwa*-gaisha (telegraph telephone company)" .

- **Sistring (semi-infinite string):**
  Text strings can be viewed as a single array of characters. A " sistring" is a subsequence of characters from this array, taken from a given starting point but going on as necessary to the right. For example, if the original text string is " Once upon a time, in a ..." , then sistring-1 is " Once upon a time, in a ..." , sistring-2 is " nce upon a time, in a ..." , sistring-8 is " on a time, in a ..." , sistring-11 is " a time, in a ..." and so on [2] [5]. Here, -n (n is a number) denotes that the n-th character of the original text string is the starting point of the sistring.

3. CONVENTIONAL METHODS FOR PARTIAL MATCHING

A DBMS makes index files from fields in a subject DB that will be accessed by users. The index files have suitable structures such as a B-tree for high-speed matching. However, the structures work effective only with exact matching and partial matching from the beginning. Therefore, all conventional

---

[1] There is no " - " in the Japanese language. Here, we insert " - " between words for convenience of reading.

methods for partial matching from the middle, such as BF (Brute-Force) method and KMP (Knuth-Morris-Pratt) method must pass the retrieving keywords across all strings in all index records in the DB [2]. When the DB has several million records, these methods are too slow.

Some full-text search methods have been proposed for Japanese texts that generate index records from full-texts for high-speed matching, such the method using an n-character index [1] and a method using a character component table. These methods create the index file from the original text beforehand using morphological analysis. The index records are words whose pointers refer to the positions of the words in the original text. When input strings are unknown whether they are composed of a word or words[2], these methods require segmenting Japanese input strings into component words and use these words as retrieving keywords. Therefore, it seems indispensable to analyze input strings on the fly. This fact means that it is difficult for these full-text search methods to invoke many processes concurrently.

In full-text search using the PAT tree, all possible strings of a text, called sistrings (semi-infinite strings), are made from the original text strings. The PAT tree is a Patricia tree constructed over all sistrings. The PAT tree allows very efficient full-text searching.

The method proposed in this paper generates supplementary index records for each original index record similar to sistrings in the PAT tree. The supplementary index records consist of sistrings; they begin at the beginning of words in each original index record and end at the final word in each original index record. Moreover, the proposed method uses a commercial DBMS instead of the PAT tree to manage the supplementary index records.

4. GENERATING SUPPLEMENTARY INDEX RECORDS

4.1 Underlying Concept

Partial matching from the middle in a DBMS is a time and resource consuming process because it must check all character positions in each index record as to whether the retrieving keywords are included in the record or not. In contrast to this, partial matching from the beginning is fast and resource efficient process because it can be performed by traversing the index tree. Namely, if partial matching from the middle converts into partial matching from the beginning, then retrieval time becomes a much shorter and a resource efficient process is realized. The proposed method generates supplementary index records in order to convert partial matching from the middle into partial matching from the beginning. This enables us to realize high-speed partial matching using a DBMS.

4.2 Flow For Generating Supplementary Index Records

Assuming that " ABCD" (A, B, C and D are words) is an original index record, a retrieving keyword such as " ABC" or " AB" can match it partially from the beginning by " key%" (% means a wild card character), for example, " ABC%" or " AB%" . However, a retrieving keyword such as " BCD" or " BC" cannot match it from the beginning. Moreover, retrieving keywords such as " AB" & " D" or " A" & " C" can also match it partially from the beginning by " key1%key2%" , for example, " AB%D%" or " A%C%" . However, retrieving keywords such as " BC" & " D" or " B" & " C" cannot match it from the beginning. Thus, since users often input a retrieving keyword(s) whose beginning corresponds to the beginning of none of original index records in a DB, we cannot perform partial matching from the beginning i.e. high-speed partial matching using only original index records.

To overcome this, we must generate supplementary index records whose beginnings are the component words of the corresponding original index record. They can be partially matched with the retrieving keyword independently of whether its beginning corresponds to the beginning of an original index record or not.

When generating supplementary index records, high frequency strings in the DB must be omitted from the index records. Such strings yield so many records that users cannot easily find the desired results among the records extracted.

Considering the above points, figure 1 shows the flow for generating supplementary index records.

- **Step 1:**
  Each original index record is segmented into its component words by morphological analysis. In figure 1, the original index record " nippon-denshin-denwa-kabushiki-gaisha" (" nippon" , " denshin" , " denwa" , " kabushiki" and " gaisha" are Japanese words) is segmented into

[2] This situation is common in Japanese information retrieval.

" nippon" , " denshin" , " denwa" , " kabushiki" and " gaisha" . As a result of morphological analysis, " nippon" /" denshin" / " denwa" /" kabushiki" /" gaisha" (/ means a word boundary) is obtained as the segmented version of the original index record.

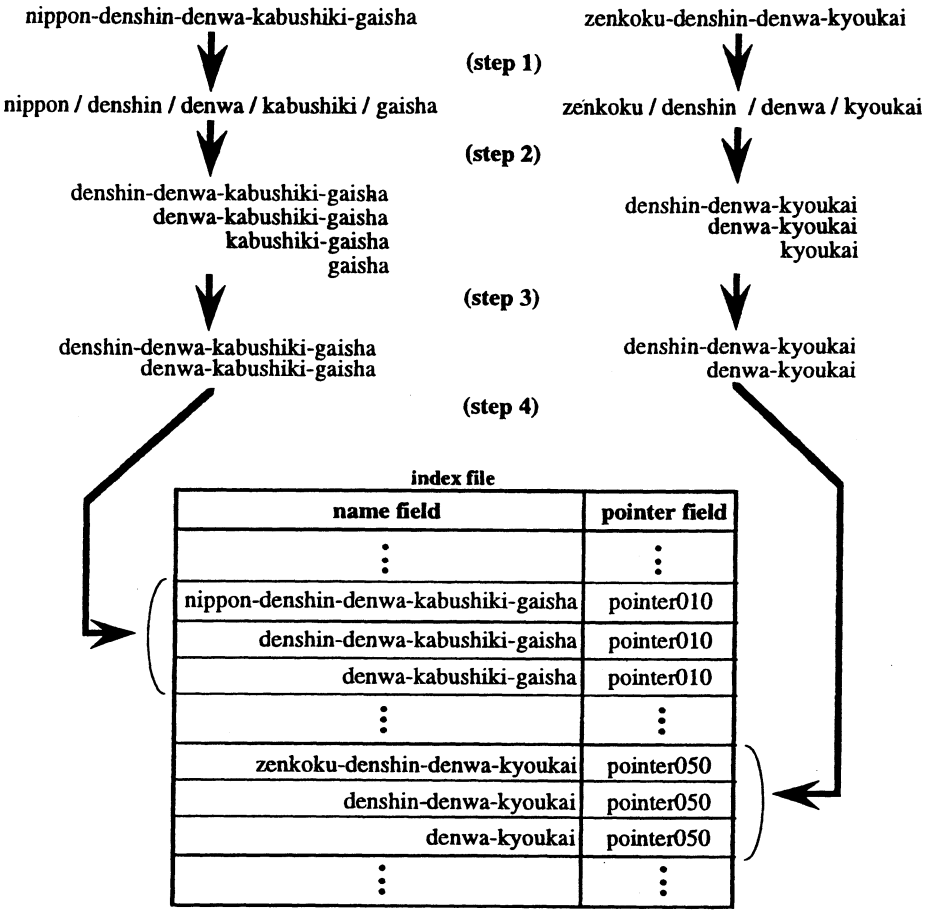

Figure 1: Flow for generating supplementary index records.

- **Step 2:**
  Supplementary index records are generated from the segmented record. The index records are made by deleting words from the beginning of the segmented original index record. In figure 1, four supplementary index records, " denshin-denwa-kabushiki-gaisha" , " denwa-kabushiki-gaisha" , " kabushiki-gaisha" and " gaisha" are generated from " nippon" / " denshin" / " denwa" / " kabushiki" /" gaisha" . In this procedure, the beginnings of all supplementary index records are at the beginnings of component words. In partial matching from the beginning, this allows us to prevent that a retrieving keyword matches a substring whose head character is at the middle of a component word.

- **Step 3:**
  High frequency records are omitted from the supplementary index records. In figure 1, assuming that " kabushiki-gaisha" and " gaisha" are high frequency records, they are omitted. Here, the frequency of each word in the DB is determined beforehand, and high frequency records are predefined.

- **Step 4:**
  Processing all original index records yields a new index file. In figure 1, for the original index record " nippon-denshin-denwa-kabushiki-gaisha" , three records (" nippon-denshin-denwa-kabushiki-gaisha" , " denshin-denwa-kabushiki-gaisha" , " denwa-kabushiki-gaisha") are registered in the new index file and all have the same pointer (" pointer010") to the corresponding information in the DB.

By applying the same procedure, for the original index record " zenkoku-denshin-denwa-kyoukai", three records (" zenkoku-denshin-denwa-kyoukai", " denshin-denwa-kyoukai", " denwa-kyoukai") are also registered and all have the same pointer (" pointer050") when assuming that " kyoukai" is a high frequency record.

5. EVALUATION EXPERIMENTS

The validity of the proposed method in terms of performance and retrieval time was confirmed in experiments.

5.1 Experimental Methods

5.1.1 Experiment 1 (for performance)

Using a commercial DBMS, Oracle 7.3.2.2, we measured three factors (described below) for the following three matching methods:

(a) The simple partial matching method: partial matching from the middle by " %key%" (index records not used).
(b) The proposed method: partial matching from the beginning by " key%" for supplementary index records.
(c) The context option: full-text search offered by Oracle 7.3.2.2. by " key" for supplementary index records.

The subject DB was the company name DB for Sapporo city, which is used in the fully automated Japanese directory assistance system [4]. This DB has about one million index records, i.e. three hundred thousand original index records and about seven hundred thousand supplementary index records.

Four-byte long character strings were randomly generated and from among these strings we selected 100 to form the retrieving keyword set. Each keyword in this keyword set can yield many records in the subject DB in retrieval regardless of the method used.

During retrieval for the keyword set by the above three matching methods, the below three factors were measured and averaged every 5 seconds:

- The CPU usage,
- The memory requirement, and
- The amount of logical disk I/O.

We measured these factors 35 times for (a) the simple partial matching method, 65 times for (b) the proposed method and 15 times for (c) the context option.

5.1.2 Experiment 2 (for retrieval time)

Using 50 retrieving keywords of different length, the retrieval time was measured by both (b) the proposed method and (c) the context option[3]. Here, the retrieval time was defined as the interval between the start time of SQL analysis of Oracle 7.3.2.2 and the time at which all retrieved records had been fetched to the output buffer from the working memory or the disk.

The subject DB for experiment 2 was the company name DB for Hokkaido prefecture, which has about three million index records, i.e. about one million original index records and about two million supplementary index records.

5.2 Experimental Results

The experimental results for CPU usage, and the amount of disk I/O are shown in figure 2 and figure 3 respectively[4]. Here, all three methods have virtually the same memory requirements.

---

[3] In experiment 1, it became clear that the retrieval time by (a) simple partial matching takes too long, so we did not measure its retrieval time.

[4] Since both sets of plots reflect transient effects, we can ignore them.

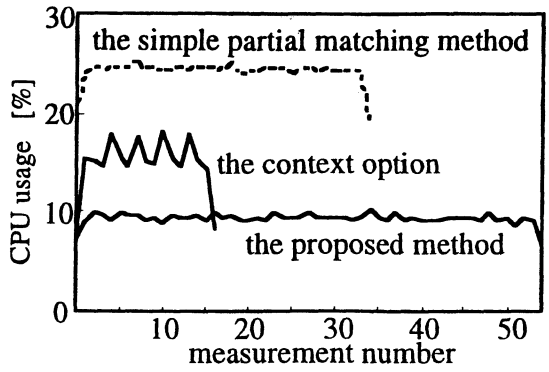

Figure 2: A result of experiment 1 (CPU usage).

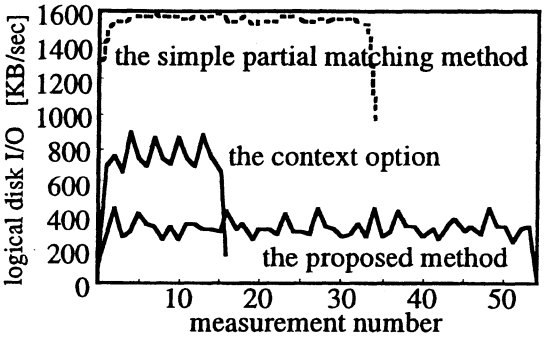

Figure 3: A result of experiment 1 (the amount of disk I/O).

(1) Experiment 1 (for performance)

Figure 2 shows that the proposed method requires about 60% less CPU power than the simple partial matching method and about 30% less than the context option. When the CPU usage is the factor dominating concurrency, this means that the proposed method can support 2.5 times more concurrency than the simple partial matching method and about 1.5 times more concurrency than the context option.

In figure 3, the proposed method requires about 80% less logical disk I/O than the simple partial matching method and about 50% less than the context option. When disk I/O is the factor dominating concurrency, the proposed method supports about 5 times more concurrency than the simple partial matching method and about 2 times more concurrency than the context option.

Namely, the proposed method satisfies one of the requirements for the telephone directory retrieval task: the system must invoke as many concurrent processes as possible.

The drawback to the proposed method is that it requires extra hard disk volume for holding the supplementary index records. However, this is a trivial problem because hard disk space has become very cheap in recent years.

(2) Experiment 2 (for retrieval time)

The result of experiment 2 is shown in figure 4 (the retrieval time vs. the amount of retrieved records). As shown in figure 4, the retrieval time of the proposed method equals that of the context option. The proposed method is slightly faster than the context option and takes less than about 5 seconds when fewer than one hundred records are retrieved.

As shown in experimental results for both experiment 1 and experiment 2 mentioned above, the proposed method can retrieve the desired results by users in several seconds while it can reduce CPU usage and disk I/O more than the conventional methods. Namely, the proposed method is practical when applied to the telephone directory retrieval task.

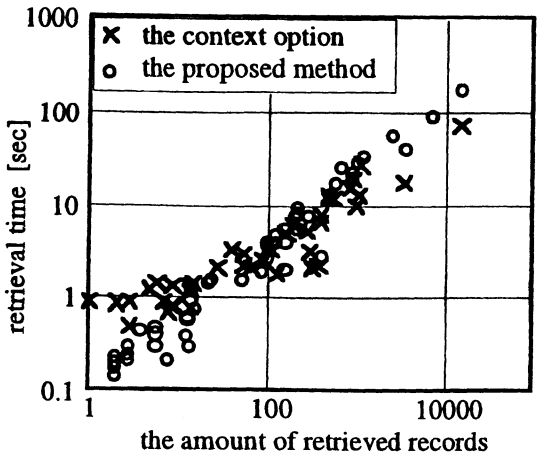

Figure 4: A result of experiment 2.

5.3 Discussion

This section discusses how to raise the precision rate for the proposed method. With the simple partial matching method, a retrieving keyword often matches a substring whose head character is at the middle of a word. Moreover, by the simple partial matching method, even by the proposed method, a retrieving keyword often matches with a substring whose end character is at the middle of a word. These facts degrade the precision rate. To avoid this problem, we proposed a method that omitted extra retrieval candidates whose beginning was not at the beginning of a word and whose ending is not at the end of a word [6].

According to the preliminary experiment on accuracy, the precision rate for simple partial matching was less than 10% while the proposed method achieved about 40%. When we checked the word boundaries as mentioned above and omitted extra candidates, the precision rate was improved to about 90%. This shows that checking word boundaries for retrieval candidates allows us to greatly improve the precision rate. The information for word boundaries is utilized for generating supplementary index records in the proposed method. Therefore, we can omit the extra retrieval candidates using this information to improve the precision rate.

It can be concluded from the above results and discussion that the proposed method is very useful against the telephone directory retrieval task because of its high-speed, reduced resource requirements, and high-accuracy.

6. CONCLUSION

This paper has proposed a method for high-speed partial matching by generating supplementary index records for each original index record like sistrings in the PAT tree. Since this enables us to convert partial matching from the middle into partial matching from the beginning, which realizes high-speed retrieval. Moreover, the proposed method can retrieve all candidates including all words within strings input by users from a subject DB without applying morphological analysis to the input strings on the fly. This shows that time and resource efficient retrieval has been realized. Experimental results showed that the proposed method satisfies the key requirements for a Japanese directory assistance system (high retrieval speed, reduced resource requirements, and high accuracy) quite well.

In terms of accuracy, only preliminary experiments have been performed. Therefore, to certify the proposed method more clearly, we will perform more extensive experiments on accuracy, i.e. precision rate and recall rate.

REFERENCES

[1] Akamine T. et al., " Flexible String Inversion Method for High-Speed Full-Text Search," Proc. of Advanced Database Symposium '96 (ADBS'96), pp.35-42, 1996. (in Japanese)

[2] Frakes W. B. et al., " Information Retrieval," Edited by William B. Frakes and Ricardo Baeza-Yates, Prentice Hall PTR, 1992.

[3] Hayashi T. et al., " Fully Automated Directory Assistance Service using a Telephone Keypad," IEICE annual meeting, D-6-5, 1997. (in Japanese)

[4] Higashida M., " A Fully Automated Directory Assistance Service that Accommodates Degenerated Keyword Input Via Telephone," Proc. of PTC'97, pp.167-174, 1997.

[5] Kikuta M., " Patricia Tree," Journal of Japan Society of Artificial Intelligence, Vol.11, No.2, pp.337-339, 1996. (in Japanese)

[6] Nagai R. et al., " Candidate Selection Based on the Character Boundary and Word Boundary," IEICE annual meeting, D-6-8, 1997. (in Japanese)