NTCIR-3 PAT Experiments at Osaka Kyoiku University
—Long Gram-based Index and Essential Words—

Takashi SATO Tomohiko SATOMOTO Koto HAN
Osaka Kyoiku University
4-698-1 Asahigaoka, Kashiwara, Osaka, Japan
sato@cc.osaka-kyoiku.ac.jp

Abstract

Long gram-based indices are experimented at NTCIR-3 patent task. To make gram-based indices, no analyses such as morphological ones are required. The docno, abj, clj and dej tag fields are extracted from NTCIR-3 patent corpus. The total index size is 11.4Gbyte and time to make indices is about 8.7 hours. Median search time per word from abj and dej index is 9.8msec and 91.8msec respectively. Average retrieval time per topic takes 73 seconds since documents are ranked in a Perl program, which is simple and not fast. Ranking algorithm used is based on a traditional probabilistic model. We also tried to set essential words in a query.

Keywords: *long gram based index, gram coding, essential word, NTCIR*

1 Introduction

Patent retrieval using computers is commonplace today in patent applications and examinations, however, NTCIR-3 patent task is novel since it regards patent retrieval as retrieval task and articles from newspaper as topics. Nowadays patent documents are accessible via internet, the chances ordinal people retrieve patents will increase more and more from now on. Specialists sometimes say that they can find words in mind well using full text retrieval systems than keyword retrieval systems. Among full text retrieval systems, systems whose indices are based on suffix array[1-5] or grams[6-10] are effective, we think, since every character sequences which include words, compound word, etc. are retrievable. In making indices, they need no dictionary and no morphological analyses.

In order to make suffix array efficiently, we have to put corpus on computer main memory. So one problem of suffix array is that we can not make indices for big corpus.

The size of corpus for NTCIR-3 PAT is 18Gbyte, which is far bigger than that of former NTCIR tasks. Since this size exceeds main memory capacity of most computers, it is impossible to make suffix array indices practically.

On the other hand, general gram-based indices are thought to have following problems.

(1) The size of index becomes huge when gram length is more than 3 for Japanese corpus.

(2) Word retrieval requires not only search for grams, which compose the word, but also computation of intersection over searched sets. Then it becomes slow.

Our group has made gram-based indices[11-13] for large corpus. We report in this manuscript that our indices do not have the above problems.

Selection of query words is very important since it directly influences precision of retrieval. It sometimes requires specialist's knowledge about target area, we think. Not only automatic systems but also interactive systems are useful for patent retrieval. As for interactive system, fast and flexible systems will be preferable.

2 Gram indices

2.1 Basics

An *n*-gram is a character sequence that has length *n*. All grams are extracted from a document in the corpus (Figure 1).

When *n* has fixed value, the index made from grams in the corpus is called *n*-gram index.

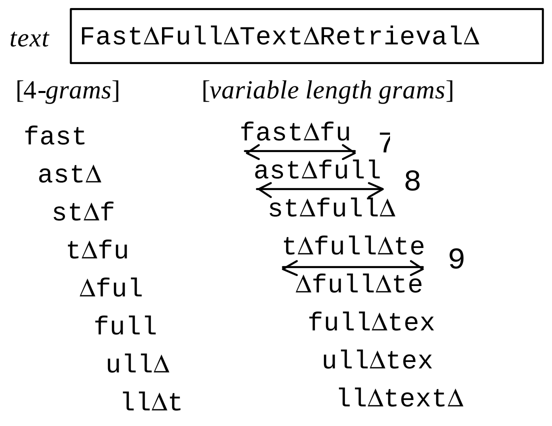

Figure 1. Grams

2.2 Variable length coding by character count

The size of *n*-gram of Japanese 2byte characters is 2*n* byte if the gram is simple concatenation of the character codes. Referencing the frequency of each character appearance in corpus, we made grams differently in order to reduce an index size. A character, whose frequency is higher, is assigned shorter code by Huffman's coding [14]. As an exception, we assign intentionally long code against space, parenthesis, punctuations and so on.

2.3 Variable length gram encoding in fixed byte

We made a gram code (gram value) from concatenation of character codes assigned as **2.2**. We fixed the length of gram value in w_g (5 or 6byte). In consequence, gram length becomes variable. Though gram values are 8byte in main memory, they are w_g on the secondary storage. When gram values stored in sorting order in the index, we express gram value in less than w_g byte by run length encoding [15] if adjacent gram values are near.

3 Index making

We compute gram values as **2**, document by document. We made an index as an inverted file of gram values. During index making, we sort gram values. We can not put the entire corpus in main memory at one time since the corpus of this task exceeds main memory size. We first made batch indices by internal sorting algorithm from subsets of corpus, which fit in main memory. Each batch is merged into an index afterward (Figure 2). In previous, we used quick sort for internal sorting. Since internal sorting dominates time, we also try radix sort [16,17] at this task.

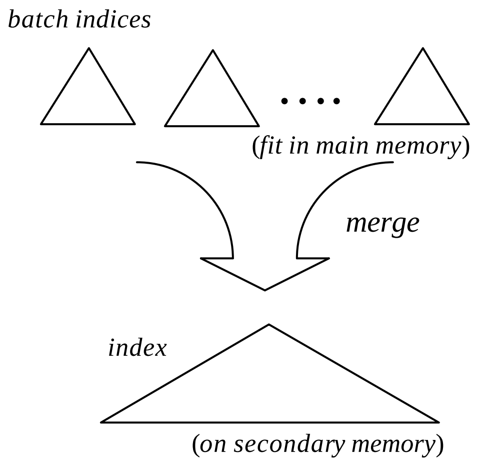

Figure 2. Index and batch indices

We also made wide range map of gram values, which are put in main memory when we search grams.

At this task, most grams constitute 4 or 5 characters, and they are coded into w_g=6 byte. That is, a gram value has almost same length as 3-gram if not coded.

Table 1 shows the size of corpus, extracted tag fields and two indices, which are made from abj and dej tag field. Index size overhead against extracted tag fields is 80.7%.

Table 1. Size of Corpus, tag fields and indices

item	size
corpus (kkh)	18.1Gbyte
<docno> tag	69.7Mbyte
<abj> tag	410Mbyte
<clj> tag	1.30Gbyte
<dej> tag	12.5Gbyte
<abj> index	101Mbyte
<dej> index	11.39Gbyte

Table 2 shows time to make indices. The radix sort version was 2.7 times faster than the quick sort version.

4 Query making

We extract query words using morphological analysis from article and supplement tags in given 31 topics (P001-31.sgml). Compound words are segmented in words, and then all possible combination of words are made of a compound word.

In this task, we set essential words. Plural words can be essential in OR, considering we can set synonyms. But we restrict them to have single idea for simplicity. From top results file, we removed documents which do not include any essential words.

5 Retrieval and results

Our index has tree structure, which has sorted gram values and wide range map of them. So, not only query words whose length is equal to gram length, but also shorter or longer words can be searched efficiently.

When we search a longer word, every gram in the words is searched. Then retrieved sets of document numbers are intersected. Each set of retrieved document number is relatively small because long gram works as a good filter. So the set operation is quite fast. Moreover, since grams are searched by gram values once they are coded, grams are searched with 8byte integer computation, i.e. high cost processing such as bit shifts or string comparisons are not required.

From set of retrieved documents for query words, we compute tf-idf and similarity using probabilistic model [18]. Document ranking by using tag location information is enabled since indices are made field by field. Table 3 shows retrieval time including ranking.

We set essential words for 4 topics, i.e. P006, P016, P029 and P030. Average precisions are improved from 0.02 to 0.23 compared with no essential words case.

6 Discussions

We think target specific synonym is necessary for patent retrieval. As for essential words, it is advisable that specialists of target areas set then since their setting are delicate. We think interactive retrieval systems effective for patent retrieval.

7 Conclusions

We experimented our long gram based indices at NTCIR-3. Since our indices are based on grams, no analyses such as morphological ones are required. We made indices from abj and dej of NTCIR-3 documents. The total index size is 11.4Gbyte and time to make indices is about 8.7 hours. Average retrieval time per topic takes 73 seconds. Ranking algorithm used is based on a traditional probability

Table 2. Time to make indices

	quick sort version		radix sort version	
field	<abj> index	<dej> index	<abj> index	<dej> index
time	44min	1,365min	20min	501min
total time	23.5hr		8.7hr	

Table 3. Retrieval Time

	<abj> index	<dej> index
retrieved words (included compound words)	151	151
time to search all words	13.7sec	122sec
search time per word (average, median)	(90.7msec, 9.8msec)	(808msec, 91.8msec)
retrieval time per query (inc. doc. ranking)	73sec	

model. We also tried to set essential words in a query.

References

[1] Gonnet, G., Baeza-Yates, R. and Snider, T., New Indices for Text: Pat Trees, in *Information Retrieval: Data Structure & Algorithms* chapter 5, Frakes, W. and Baeza-Yates, R. Ed., pp. 66-82 (1992).

[2] Shang, H. and Merrett T., Trees for approximate string matching, *IEEE Trans. Knowledge and Data Eng.*, Vol. 8, No. 4, pp. 540-547 (1996).

[3] Itoh, M., An Efficient Method for Constructing Suffix Arrays of Large Texts, *IPS Japan SIG Notes*, 99-NL-129-5 (1999).

[4] Yamashita, T., Fujio M. and Matsumoto Y., Language Independent Tools for Natural Language, *Proc. 18th ICCPOL*, pp.237-240 (1999).

[5] Ferragina, P. and Grossi, R., Fast string searching in secondary storage: Theoretical developments and experimental results, *Proc. ACM-SIAM Symposia on Discrete Algorithms*, Vol. 7, pp. 373-382 (1996).

[6] Ogawa, Y. and Iwasaki, M., A new character-based indexing method using frequency data for Japanese documents, *In Proc. 18th ACM SIGIR Conf.*, pp. 121-129 (1995).

[7] Sugaya, N. *et al.*, A full-text search system for large Japanese text bases using n-gram indexing method, *Proc. 53th Annual Convention IPS Japan*, 5T-2,3 (1996).

[8] Akamine, S. and Fukushima, T., Flexible string inversion method for high-speed full-text search, *Proc. Advanced Database Symposium '96* (1996).

[9] Matsui K., Namba, I. and Igata, N., Full-text searching engine for large-scale data, *Proc. 1997 IEICE General Conference*, D-4-6 (1997).

[10] Kikuchi, C., A fast full-text search method for Japanese test database, *Trans. IEICE*, Vol. J75-D-1, No. 9, pp. 836-846 (1992).

[11] Sato, T., Fast full test search with free word using TS-file, *Proc. 19th ACM SIGIR Conf.*, p.342 (1996).

[12] Sato, T., Fast full test retrieval using gram based tree structure, *Proc. ICCPOL '97*, Vol.~2, pp. 572--577 (1997).

[13] Sato, T. *et al.*, Gram based full test search system and its application, IPSJ SIG Notes, 98-DBS-114-2 (1998).

[14] Huffman, D.A., A method for the construction of minimum-redundancy codes, *Proc. IRE*, Vol. 40, pp. 1098-1101 (1952).

[15] Zobel, J., Moffat, A. and Sacks-Davis, R., An efficient indexing technique for full-text database, *Proc. 18th Int. Conf. on VLDB*, 1992, pp. 352-362.

[16] Knuth, D., *The Art of Computer Programming: Vol.3 Sorting and Searching, 2nd Ed.*, Addison-Wesley, Reading, Mass., pp. 481-489 (1998).

[17] McIlroy, P., Engineering radix sort, *Computing Systems*, Vol. 6:1, pp. 5-27 (1993).

[18] Robertson, S.E. and Walker, S., Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval, *Proc. 17th Int. Conf. Research and Development in Information Retrieval*, pp. 232-241 (1994).