A Distributed Retrieval System for NTCIR-5 Patent Retrieval Task

Hiroki Tanioka Kenichi Yamamoto
Justsystem Corporation
Brains Park Tokushima-shi, Tokushima 771-0189, Japan
{hiroki_tanioka, kenichi_yamamoto}@justsystem.co.jp

Abstract

We developed a distributed search system with the corresponding very large scale corpora from NTCIR-5 Patent Retrieval Task. And we developed the method of query refining using Support Vector Machines. Our search system, which consists of 5 PCs could make indices of all claims for ten years. Additionally, we confirmed that our arranging the scoring method made an improvement of mean average precision.
Keywords: *distributed information retrieval, support vector machines, vector space model, inverted file*

1 Introduction

Our purposes to participate in NTCIR-5 Patent Retrieval Task is as follows.

- Research and development search systems which are corresponding a very large scale corpora.
- Research and development query refining methods which are useful for "invalidity search"

The background of first purpose is that digital documents are increasing in recent years, while we need search systems to effectively access these documents. But traditional search systems cannot make the full text indexes for these documents. Therefore we propose a distributed search system which is build on a distributed framework.

A background of second purpose is that it is high cost to make query from claims manually for invalidity patent search. Then, we try to make query from claims automatically.

The rest of this paper is divided into three sections. Section 2, we describe an architecture of our distributed processing framework and search system. Section 3, we describe results of formal runs. Section 4, we discuss about results and future works.

2 System Description

In this section we describe the architecture of our distributed search system and information retrieval models including some scoring methods.

2.1 Document Retrieval Subtask

First, we explain the system architecture and models for Document Retrieval Subtask.

2.1.1 Overview

We develop a distributed search system which is based on Vector Space Model using term partitioning with an inverted file-based system, while a single inverted file is created for the document collection and the inverted lists are spread across the processors.

During query evaluation, the query is decomposed into indexing items and each indexing item is sent to the processor that holds the corresponding inverted list[1].

2.1.2 Distributed Processing Framework

Cocktail Framework[1] is used to make the distributed search system based on Vector Space Model. The framework provides a service of agents between client and server as broker between query and each indexing item.

Figure 1 shows an overview of this framework. To process a job[2] in our system, a client machine receive a job, and keep in a FIFO queue. And then, to send the job to a server machine, unconfined agents pull the job from the FIFO queue. Last, the server machine performs the job and send a result back to the client machine via same agent.

[1] Cocktail Framework is developed for distributed processing framework by Justsystem Corporation.

[2] Job is described as a pair of command and argument which are processed in our system.

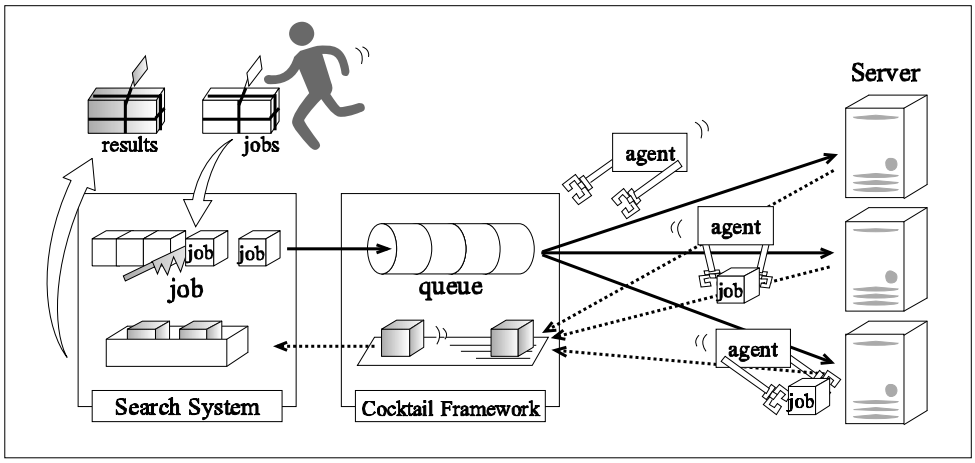

Figure 1. System Description

2.1.3 Indexing Algorithm

This system has indexing structure using an inverted file. And this inverted file-based system is based on term partitioning for whole distributed inverted files on some server machines as a single inverted file.

In general, there are two methods of how to distribute inverted files on server machines. First method is that an inverted file is divided based on terms. Second method is that an inverted file is divided based on documents. And, the second method needs to search the inverted file on all server machines for the documents, but it is hard to calculate the correct IDF scores for each terms in a query. Thus we decide on to apply the first method.

The problem is that it is too costly to make the inverted file. It needs a large amount of memory if we execute on memory, and it needs a long time if we execute on hard disk. Therefore we take an approach using a kind of merging the partial indices increasingly. In concrete, we show the conceptual figure as Figure 2. We make the inverted file on memory until reach a limit. When the size of inverted file is over the limit of memory, we store the inverted file on memory to hard disk. If there are already a previous inverted file on hard disk, the two files on hard disk and on memory are gradually merged.

The total time to generate the partial indices is $O(n)$, where n is the number of characters, m is the number of merging times. And thus the cost of this algorithm is as follows,

$$O(\frac{n' \cdot m(m+1)}{2}) \simeq O(n' \cdot m^2) \qquad (1)$$

where n' is an average number of characters each partial indices.

2.1.4 Retrieval Model

Our retrieval model is based on Vector Space Model. And calculating formula of search score is based on simple calculation of $TF \cdot IDF$ as follows.

$$S_Q = \frac{p_1}{T_q} \qquad (2)$$

$$S_T = \log(TF) \cdot \log(\frac{N}{DF}) + S_Q \qquad (3)$$

Where S_Q is the score dependent on the number of terms T_q in given query, S_T is the score of term t, TF is term frequency, N is the number of documents, and DF is document frequency. To prove the limitation of $\log(TF)$, we use p_2 instead of $\log(TF)$ when $\log(TF)$ is greater than p_3.

Here in the experimentations, the values of parameters are set without making an adjustment at all. Where each constant numbers are declared as follows.

$$p_1 = 100, \quad p_2 = 1, \quad p_3 = 2 \qquad (4)$$

There are three differences from original $TF \cdot IDF$ calculating formula.

- Using logarithm for term frequency : TF
- Addition of score : S_Q
- Limitation of logarithmic TF : $\log(TF)$

The first difference is based on our pilot study, which shows an adverse effect a greater values of TF on original $TF \cdot IDF$ calculation. The second difference is that operationality method controls the score based on the number of terms in a query. And a purpose of the third difference is a solution of detecting high frequency terms.

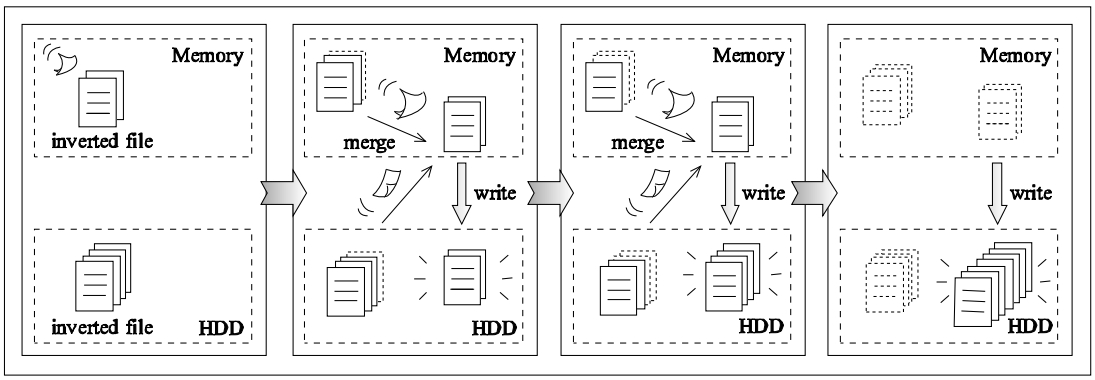

Figure 2. Indexing Algorithm

In addition, the feature of Vector Space Model contains terms of noun, verb and unknown word as part of speech from all claims by using a tool of morphological analyzer[3]

2.1.5 Query Processing

It is the high cost to make queries from claims manually for invalidity patent search. Then we try to make queries from claims automatically. In this instance, we use Support Vector Machines (SVMs)[6, 3] to choose terms from a claim as a query. And we use LIBSVM[2] as a library for SVMs.

We prepare a training data of 40 queries by patent administrator in Justsystem Corporation. The features for SVMs are surface and part of speech as follows.

- Word of current (Current Word)
- Word of previous (Pre Word)
- Word of next (Post Word)
- Self-sufficient word of previous (Pre Self Word)
- Self-sufficient word of next (Post Self Word)

And, Figure 3 shows an example of features. Where Self-sufficient word usually said in Japanese is subequal to content word in English.

2.2 Passage Retrieval Subtask

The search system for Passage Retrieval Subtask is the same system of Document Retrieval Subtask with some exceptions as follows.

- The system for Passage Retrieval Subtask consists of 1 PC only.
- One passage is defined as one document to use the system for Document Retrieval Subtask.

[3]The tool of morphological analyzer was developed by Justsystem Corporation features Hidden Markov Model and Bigram.

3 Results

In this section we show the result of NTCIR-5 Patent Retrieval Task.

3.1 Document Retrieval Subtask

It shows the result of Document Retrieval Subtask. Table 1 shows the difference of 2 runs we submitted.

Table 1. Difference of each runs

	variety of query
JSPAT1	all terms of query
JSPAT2	selected terms using SVMs

Table 2 shows the performance of our system. The indexing time was without the morphological analysis time.

Table 2. Performance of Search System

Indexing time	8.81 hours
Search time (JSPAT1)	40.05 sec.
Search time (JSPAT2)	9.86 sec.

Table 3 shows that specification of these PCs. And expanded information is that each network-linked PCs are connected on gigabit Ethernet.

Table 3. Specification of PCs

	CPU[GHz]	Ram[GB]	OS
A	Celeron 2.4	1	WinXP Pro SP2
B	Celeron 2.4	1	FedoraCore3
C	Celeron 2.4	2	FedoraCore3
D	Celeron 2.2	1	FedoraCore3
E	Celeron 2.2	1	FedoraCore3

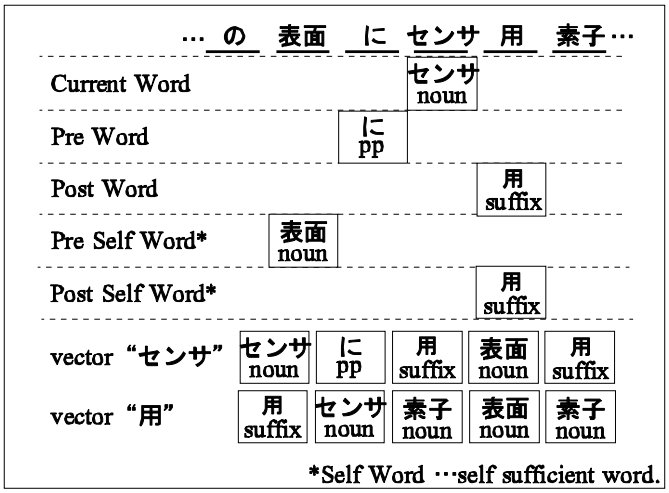

Figure 3. Features of SVMs

On that basis, the time for indexing all claims was 8.81 hours. And the time to search for invalidation of patents was 9.86 or 40.05 second for each query. The variation in search time was caused by the difference in the amount of terms in a query. As a result, we thought that the distributed search system showed a significant advantage for the processing time.

Table 4 shows mean average precisions result of all runs. Where others is an average of the other participants' mean average precisions.

Table 4. Result of Each Runs

	ntcir5a	ntcir5b	ntcir4a	ntcir4b
JSPAT1	0.0667	0.0532	0.0912	0.0588
JSPAT2	0.0587	0.0484	0.1085	0.0768
others	0.1592	0.1286	0.2021	0.1795

According to these results, the result of JSPAT2 was better than JSPAT1 on ntcir4a and ntcir4b. But JSPAT2 was worse than JSPAT1 on ntcir5a and ntcir5b. We cannot say that selected terms using SVMs lead to a positive outcome.

3.2 Passage Retrieval Subtask

We show the result of Passage Retrieval Subtask. Table 5 shows the difference of 2 runs. JSPAT1 was submitted for the formal run. And JSPAT2 was the result of our optional experiments.

Table 5. Difference of Each Runs

	variety of query	submitted
JSPAT1	all terms of query	Yes
JSPAT2	selected terms using SVMs	No

Table 6 shows mean average precisions. Where others is an average of the other participants' scores. Then we could confirm that our results were positive in mean average precision.

Table 6. Mean Average Precision

	a.a	a.b	b.a	b.b
JSPAT1	0.5228	0.4785	0.4900	0.4626
JSPAT2	0.4431	0.4146	0.4244	0.4092
others	0.3404	0.3433	0.3256	0.3320

And Table 7 shows precision oriented scores. We could confirm that our results were also positive in precision oriented score.

Additionally, JSPAT1 was slightly batter than JSPAT2 in mean average precision. Meanwhile JSPAT1 was slightly worse than JSPAT2 in precision oriented score. It is easy to assume that the reduction of terms in a query causes the recall to decrease and the precision to increase.

Table 7. Precision Oriented Score

	precision oriented score
JSPAT1	11.67
JSPAT2	10.86
others	16.20

The system for Passage Retrieval Subtask has the two characteristics different from the system for Document Retrieval Subtask. The first feature is just a difference in the size of the corpus. However the second feature produces a structural difference because one passage is defined as one document to use the system for Document Retrieval Subtask.

4 Discussion

We proposed the distributed search system and indexing method using merging the partial indices. Also we evaluated the distributed and indexing methods, and showed that these methods have high-speed in Document Retrieval Subtask. Therefore we can say that restriction of corpus size is solved in theoretical.

We also proposed to use the selected terms using SVMs. When this term refinement method was used, the results was less than improved. However the search time for JSPAT2 was quad-time of JSPAT1. Additionally the distributed processing framework and search system, which were very flexible to adjust to various experimental environments.

All our purposes are accomplished, but there are unsatisfactory results. We show the calculating formula again.

$$S_T = \log(TF) \cdot \log(\frac{N}{DF}) + S_Q \qquad (5)$$

Here in the above formula, there are problem that $\log(TF) = 0$ when $TF = 1$. Thus we improved the calculating formula as follows.

$$S_T = \log(TF + 1) \cdot \log(\frac{N}{DF}) + S_Q \qquad (6)$$

Table 8 shows the results improved calculating formula. When all experimental conditions are same as original conditions, new results were better than all original results.

Table 8. Improved Result of Each Runs

	ntcir5b	ntcir4b
JSPAT1	0.0532	0.0588
new JSPAT1	0.0739	0.1013
others	0.1286	0.1795

However the calculating formula is written in a component of search system. Then we can these additional experiments without re-indexing the inverted file.

Finally our future work is the improvement of search result. We must experiment various calculating formula and parameters. Although we will optimize quickly, we can simply change components. In addition, we should try to refine the query for claims using any other machine learning methods.

Acknowledgement

We appreciate our colleagues Kayoko Tonoi and Daisuke Motohashi for their encouragement.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*, chapter 5-9. Addison-Wesley, 1999.

[2] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[3] J.-T. N.Cristianini. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[4] D. G. S. Richard O. Duda, Peter E. Hart. *Pattern Classification Second Edition*, chapter 5.11. John Wiley & Sons Inc, 2000.

[5] Y. C. S. Salton G., Wong A. A vector space model for automatic indexing. *Communications of the ACM*, 613-620(18), 1975.

[6] V.N.Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

[7] J. Wu, H. Tanioka, S. Wang, D. Pan, K. Yamamoto, and Z. Wang. An improved vsm based information retrieval system and fuzzy query expansion. In *FSKD (1)*, pages 537–546, 2005.