Study on Merging Multiple Results from Information Retrieval System

Hiromi itoh OZAKU†‡, Masao UTIYAMA†, Hitoshi ISAHARA†
†Communications Research Laboratory
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan
{romi, mutiyama, isahara} @crl.go.jp

Yasuyuki KONO‡, Masatsugu KIDODE‡
‡Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-city, Nara, 630-0101, Japan
{kono, kidode}@is.aist-nara.ac.jp

Abstract

We participated in Web retrieval tasks in order to investigate the integration of multiple retrieval results. We also examined the possibility of using a general retrieval system designed for organized documents, such as news articles that match users' queries, to retrieve unorganized documents, such as Web documents.

Our results show that if information retrieval systems use the OKAPI method, results can be used for Web retrieval that is based on scored results without score normalization. On the other hand, if the systems use Rocchio's feedback method, ranked results are shown to be the best.

Keywords: *OKAPI, Rocchio's formula, Web retrieval, meta-search*

1 Introduction

In general, information retrieval systems use one index file made by each system in order to calculate the similarity between the data and the queries. It is possible to generate one optimal index file by using documents from the whole Web in the Internet. However, the number of documents on the Internet has increased dramatically. Collecting and managing Web documents with one machine has become much more difficult therefore multiple systems have been used in retrieving Web documents.

The multiple systems are called meta-search systems. The conventional meta-search systems need to combine the returned lists of pages from different search engines. At that time, the systems either directly show the results with the name of the specific search engine or present new results that have been recalculated by using various normalizing methods [7] [2].

Generally, score normalization methods are adopted by the meta-search systems. However, the normalization process is considered to be a time-consuming task and is sometimes unrealistic. Our interest is to find efficient methods that do not use the score normalization process, in order to retrieve Web documents rapidly.

We examined three merging methods without score normalization and one normalizing method in Web tasks. The merging methods depend on the outputs ranking from the information retrieval systems, using both scores and ranking, or only scores. Results showed that using the OKAPI search method without automatic feedback, could be efficient for Web retrieval without a score normalization process. In addition, our scored results showed an equivalent level to the score normalization results. On the other hand, Rocchio‘s feedback method showed the best-ranked results.

In this paper, we discuss the merging methods and examine the results we obtained by submitting Web tasks at NTCIR3. In our approach, the key points are not only the precision of each search engine among the multiple systems, but also the way to merge results from each system in order to produce the best overall results.

2 Information Retrieval System

We used an IR package [8] [1] which is opened to the public as a general retrieval system. This package is considered as a preprocessing tool for natural language processing tasks, as well as an information retrieval system. It has two functions: the first is to retrieve documents that match a query, while the second

[1] IR package can be downloaded on http://www.crl.go.jp/jt/a132/members/mutiyama/software.html. We used the IR package version 1.47.

is to collect statistics on the occurrence frequencies of certain words. In this paper, we focus on the retrieval function.

2.1 Scoring Function

The IR package uses the Okapi BM25 function [6], which is as a probabilistic-type function. When a query Q is given, the score of a certain document D is calculated using the following equation:

$$\sum_{T \in Q} w^{(1)} \frac{(k_1+1)tf}{K+tf} \frac{(k_3+1)qtf}{k_3+qtf} \qquad (1)$$

where

Q is a query, containing terms T, and $w^{(1)}$ is the weight of T in the query Q as expressed in the following equation:

$$\log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \qquad (2)$$

Here, N is the number of documents in the collection;
n is the number of documents containing T;
R is the number of documents known to be relevant to Q;
and r is the number of documents relevant to Q and containing T. [2].
Returning to equation (1), K is expressed as $k_1((1-b)+b\frac{dl}{avdl})$.
k_1, b and k_3 are constants set according to previous results. These constants were set with default value of $k_1 = b = 1$ and $k_3 = 1000$.
dl is the length of D as a substitute for the number of terms contained in D.
$avdl$ is the average document length in the collection.
tf is the occurrence frequency of T in D.
Finally, qtf is the occurrence frequency of T in Q.

The retrieval results are ordered by the document score based on equation (1).

2.2 Automatic Feedback

Equation (1) implements a retrieval function which utilizes a feedback process as adapted in equation (2). In the IR package, another feedback retrieval function which based on a type called Rocchio's feedback[3] is implemented.

[2] R and r are fixed by either the user or the IR system's estimation. In this package, R and r were initially taken as 0 for convenience.

Using Rocchio's feedback, R and r both have values of 0 in equation (2). Then qtf in equation (3) becomes

$$qtf = \alpha qtf_0 + (1-\alpha)\frac{\sum_{i=1}^{R} qtf_i}{R} \qquad (3)$$

where

qtf_0 is the occurrence frequency of a term corresponding to the initial query.
qtf_i is the occurrence frequency of a term contained in the documents ranked among the top i of the retrieval documents.
R is the number of documents utilized for automatic feedback. The default value is $R = 5$
α is a constant set according to previous results, with a default value of 0.5.

For the Web retrieval task, we submitted the results of the OKAPI retrieval, called the normal method (**N**) and the Rocchio's feedback retrieval, called the feedback method (**FB**).

3 Hardware

We used a Linux system at the open laboratory of the National Institute of Informatics (NII) to examine all the processes within our tasks. The Linux machine had the following characteristics: two 1-GHz Pentium III CPUs, 1 GB of memory, and 1.2 GB of swap. The Linux OS was Red Hat 7.1.

4 Web Retrieval Process

We submitted our results to the following tasks: A1 Topic Retrieval, A2 Similarity Retrieval, and B2 Target Retrieval. Two document collections of 100 Gbytes (**LARGE**) and 10 GB (**SMALL**) were used, with both raw and cooked data provided by NII. The cooked data were pre-converted to EUC code, and non-alphanumeric data such as graphics, movies, and the like were eliminated. The resulting size of the LARGE cooked data was approximately 39 GB, and the size of the SMALL cooked data was about 4.8 GB. The number of documents in the LARGE collection was 11,038,720, while the number in the SMALL collection was 1,445,466.

Our process in using the IR package is listed as follows:

- Transform the cooked data to formatted data by using the IR package.
- Generate index files from the resulting formatted data by using the 'irbuilder' command of the IR package.

```
<NW:DOC>
  <NW:META>
    <NW:DOCID>document ID</NW:DOCID>
    <NW:URL>URL of the page</NW:URL>
    <NW:DATE>gathered date</NW:DATE>
    ...
  </NW:META>
  <NW:DATA>
    <NW:DSIZE>data bytes</NW:DSIZE>
    page contents
    </NW:DATA>
</NW:DOC>
```

Figure 1. NTCIR cooked data format

```
<DOC NAME=''document ID''>
<TITLE>document ID''</TITLE>
<DATE>gathered date<DATE>
<TEXT>
  page contents
</TEXT>
</DOC>
```

Figure 2. IR package format

- Retrieve results for all queries of the Web tasks.
- Transform the results to NTCIR form.

Each process had some problems. In the following sections we discuss each process in detail and explain our proposed solutions to each problem.

4.1 Transforming the Data

A format of the cooked data is shown in Figure 1. The cooked data contains Web machine information in the META fields, and html file data in the DATA fields. The html file data were pre-converted to EUC code, non-alphanumeric data such as graphics, movies, were eliminated.

The page contents in the cooked data included many space codes and control codes, which we removed, except for Kanji codes. We then transformed the NTCIR cooked data into the IR package format, as shown in Figure 2. The IR data has the same document ID as the cooked data in DOC NAME fields and text information in TEXT fields that were transformed by eliminating html tags that did not have related contents, for example font size information, table line information and the like.

In this step, many errors were generated. We did not analyze all the errors completely, but we considered them as related to the existence of files with incorrectly converted code or file-size limitations that were processed with the IR package. These errors might also be related to hardware limitations.

We next eliminated cooked files and documents according to the following criteria:

warning: **NW001696888** of 011965: characters in a line over 25000(32000)
warning: **NW001697123** of 011965 is over 1,000,000 (1017805)
warning: **NW001607100** of 011195: characters in a line over 25000(517807)
warning: **NW001966687** of 013887: characters in a line over 25000(46867)
warning: **NW002103675** of 014826: characters in a line over 25000(517807)
warning: **NW002707583** of 019598: characters in a line over 25000(30337)
warning: **NW002361678** of 016731 is over 1,000,000 (4675648)
warning: **NW002475775** of 017593 is over 1,000,000 (2079216)
warning: **NW001918225** of 013585 is over 1,000,000 (1687263)
warning: **NW002180204** of 015447 is over 1,000,000 (1012667)
warning: **NW003174089** of 022989 is over 1,000,000 (1664762)
warning: **NW003784469** of 027129: characters in a line over 25000(84316)
warning: **NW003784470** of 027129: characters in a line over 25000(84316)
warning: **NW003991386** of 028573: characters in a line over 25000(98940)
warning: **NW003991400** of 028573: characters in a line over 25000(27275)
warning: **NW003991694** of 028573: characters in a line over 25000(27275)
warning: **NW000224079** of 001359 is over 1,000,000 (1175805)
warning: **NW001137337** of 008309: characters in a line over 25000(26080)
warning: **NW001175481** of 008581: characters in a line over 25000(69387)
warning: **NW006942200** of 045893: characters in a line over 25000(30962)
warning: **NW008601480** of 055641 is over 1,000,000 (1869815)
warning: **NW010462950** of 068046 is over 1,000,000 (1540714)
warning: **NW010768429** of 070084: characters in a line over 25000(26758)
warning: **NW010768432** of 070084: characters in a line over 25000(26758)
warning: **NW011177333** of 072867: characters in a line over 25000(60480)
warning: **NW011594459** of 075256: characters in a line over 25000(27858)
warning: **NW013419684** of 088115: characters in a line over 25000(47181)
warning: **NW014794137** of 097628: characters in a line over 25000(31028)
warning: **NW014401503** of 094677: characters in a line over 25000(44163)
warning: **NW014896431** of 098493: characters in a line over 25000(37010)
warning: **NW014896431** of 098493: characters in a line over 25000(72044)
warning: **NW014896431** of 098493: characters in a line over 25000(51858)

Figure 3. List of files eliminated from the SMALL Collection

- Cooked data files of more than 10 MB;
- documents with over 25,000 characters in one line;
- and documents of more than 1 MB.

Ultimately, we eliminated 32 documents, as shown in Fig. 3, and three cooked files (000046, 008792, 096956) from the SMALL collection.

From the LARGE collection, 430 documents (NW001696888, etc.) and 170 cooked files (097775, etc.) were eliminated. This reduced the size of the SMALL collection to approximately 2.4 GB, and the size of the LARGE collection to about 21 GB. Thus, the number of documents in the SMALL collection became 99% of the number of documents in the NTCIR produced data, while the number of LARGE collection documents became 94% of the number of documents in the NTCIR produced data.

4.2 Generating Indexes

In this step, we encountered one problem related to the size of the index file.

In general, information retrieval systems use one index file that is made by each system, in order to calculate the similarity between the data and the queries. If information retrieval systems can create one index file by using the whole collection of documents in the Internet, it could be said that the index file is the optimal index file.

However, it is difficult to manage huge documents with just one index file due to the limitation of the IR package. On the other hand, search engines for Web documents currently employ multiple machines, in order to cover a wider range of the Internet or search more efficiently. The multiple machines are dispersed at some places for load sharing, trouble prevention and the like. In this situation, it is difficult to make one index file and share the one index file with the multiple machines that are established at different places. We should merge multiple outputs from each machine and make one result.

Assuming a distributive situation, we set the input file size to 500 MB for the SMALL collection and 1 GB for the LARGE collection. We then generated five index files for the SMALL collection and 21 index files for the LARGE collection.

Index files were not generated when the line length of the input file was too long. We set a limitation of 8,000 terms for one line length and eliminated documents with line lengths larger than this limitation.

As a result, 129 documents were eliminated from the SMALL collection, while 494,687 documents were eliminated from the LARGE collection [3]. Ultimately, we had 1,445,267 documents in the SMALL collection and 9,902,195 documents in the LARGE collection. The number of documents in the LARGE collection decreased to 89% of the number in the NTCIR produced data.

4.3 Retrieving Results for All Queries

We used three types of queries: those using only the description field DESC (called **D**), those using only the TITLE field (**T**), and those using the TITLE field and the first relevant documents field RDOC[1] (**TRD**). The DESC field provides a basic description of user request, the TITLE field is composed of one to three words that represent the essence of the user request. The RDOC field identifies three relevant documents, RDOC[1] means the first relevant document in the RDOC fields.

[3]The documents eliminated from the LARGE collection included all the documents (493,709 documents) in one particular input file. We tried various ways to generate an index file from this input file but were unable to do so during the formal run term. Thus, we used 20 index files for the LARGE collection.

In general, the so-called meta-search engines combine the results of multiple search engines. In this case, how do meta-search engines merge and sort the scores from each search engine? Conventional meta-search systems combine the returned lists of URLs or pages from different search engines. At that time, the systems either directly show the results with the name of the specific search engine or present new results that have been recalculated by using various normalizing methods [7] [2].

For this task, we used the scores and rankings produced by the IR package output. We examined four methods for using the combined results. All four methods worked automatically.

The first method, called **Score**, merges some outputs depending on the scores. It organizes the results which are based on the IR package's output scores for each index file.

The second method, called **Each**, merges some outputs using the scores and ranking. It consists on two steps. First, the results are extracted for N ranked documents from all output over 1,000 documents. Second, the final results are organized according to the output score. For example, for the SMALL collection we made five index files. As a result, we obtained five outputs, extracted the top 200 documents from each output file, and combined them into one file as a final result according to the score of each document.

The third method, called **Top**, merges multiple outputs by the ranking. It includes several steps. The first step is to extract the first-ranked document from each output file. The next step is to order the extracted documents according to their scores and define the ranked documents as a result. We repeat this process for the second-ranked document from each output file. These steps are continued until we have over 1,000 documents in the final result. For example, in the SMALL collection, we had five output files, so we extracted the five documents that ranked first in each output, ordered them from first to fifth according to their scores, and defined them as the top five documents of the final result. Next, we extracted the five documents that ranked second, ordered them by rank, and defined them as the sixth through tenth documents of the final result. We then continued this process until we had over 1,000 documents in the final result.

Finally, the third method, which uses score normalization, is called **SN**. This method consists of rebuilding $avdl$ and n from all index files just the same as making one index using the whole data collection. We believed the best way was to make one index using the whole data collection, then retrieve information using the OKAPI method. However, the system is not able to make one index using the whole data collection because of the limitations of the machine, file size, text length and the like. Therefore, we recalculated $avdl$ and n from the set of indexes, as if we had made

one index. The OKAPI score is obtained for the SN method.

5 Experimental Results

We obtained three official results from NII as EVF, DCG[5], and MRR[9]. The EVF results were created by 'trec_eval' [1], which was used to evaluate the official TREC results. We describe the EVF results in detail.

5.1 Survey Retrieval

Tables 1 and 2 show the experimental results for the SMALL and LARGE collections, respectively. In the tables, we classified the results into four classes depending on query length and method. Bold numbers show the best score for each class.

The label **qp-cont** refers to the results when the assessor judges the relevance so that high relevance, relevance, and partial relevance are wholly judged as the same level of relevance, with consideration of only the content of the document itself. The label **qp-link** refers to the results when high relevance, relevance, and partial relevance are wholly judged as the same level, with consideration of both the content of the document and the out-linked documents on the condition that they are included in the document pools. The label **q-cont** refers to the results when high relevance and relevance are judged as the same level, with consideration of only the content of the document itself, while **q-link** refers to using the content of the document and the out-linked documents in the document pools.

We expected that the score normalization method (SN) would have an effect on retrieving relevant documents, and also on the ranking, with a high rank for a strongly related document. We also expected that the Rocchio's feedback retrieval method (FB) would be better than the method without feedback that was method N.

The official review, however, was contrary to our expectations. When the query was only DESC (D) in the SMALL collection, the combination of scores only and method N produced the best score for qp-cont, qp-link, q-cont and q-link. When the query was D in the LARGE collection, almost the same results were obtained.

On the other hand, when the feedback method is used, the ranked method (TOP) got the best score in spite of the collection size.

In conventional information retrieval systems, better results are always generated by long queries with several terms generally. Thus, when the query is TRD, the combination of the Score method and method N achieves the best score in the SMALL collection. In the LARGE collection, we obtained similar results.

When comparing the three methods of Score, Each and Top, the combination the Score method and method N produces the best score in both collections. On the other hand, the Top method produces the best score in both collections on the method FB.

Comparing methods N and FB, we found that method N tended to achieve better scores. This tendency is especially strong for short queries, which indicates a weak point for method FB. When feedback documents include non-relevant documents, the results were the worst. If we use the interactive mode to select the feedback documents, method FB should produce better scores.

In the automatic mode and the OKAPI method, we consider that the Score method is the best approach that is equivalent to SN method. In the feedback method, the Top method is the best way.

5.2 Target Retrieval

Among the results for R-Precision in the SMALL (Table 3) and LARGE (Table 4) collections, TRD produced the best score for P@5 and P@10 The P@5 and P@10 mean the precision at five and ten documents retrieved that produced by the evaluation program 'trec_eval' [1].

In the SMALL collection, the average R-Precision score for the Score method combined with method N showed the best results. In the LARGE collection, the Score and Each method obtained the best score for P@5 and for the average in qp-link. For this subtask, combining the Score method and method N is the most efficient approach in the automatic mode, especially for long queries. For method FB, the Top method has the best score.

The reason for the efficiency of the Score method is considered to be as follows: If each index file includes the same contents, the relevant results obtained from all index files become similar. This means results can be obtained with similar scores. The relevant documents get higher scores, and then the highly relevant documents extracted from all index files can be merged in the order of the highest to the lowest relation using the Score method.

In general, when systems collect Web documents and distribute them to index files automatically, the contents of the index files are distributed uniformly. In such a situation, the Score method is very effective and equivalent to the SN method.

However, when the index files are made or classified according to the contents in special subjects, it will be effective to select an index file that is suitable for the query. On the other hand, when the search system uses method FB, the Top method is very effective.

Table 1. Results for Average Precision (SMALL)

No.	Query	Method Used				Average Precision			
		Score	Each	Top	SN	qp-cont	qp-link	q-cont	q-link
1	D	N				**0.1844**	**0.1559**	**0.1638**	**0.1381**
2	D		N			**0.1844**	**0.1559**	0.1637	0.1380
3	D			N		0.1656	0.1392	0.1466	0.1233
4	D				N	0.1834	0.1572	0.1570	0.1349
5	D	FB				0.1352	0.1141	0.1213	0.1034
6	D		FB			0.1386	0.1192	0.1236	0.1067
7	D			FB		**0.1582**	**0.1366**	**0.1313**	**0.1117**
8	T	N				0.1702	0.1414	0.1668	0.1262
9	T		N			0.1702	0.1413	0.1667	0.1261
10	T			N		0.1549	0.1270	0.1591	0.1158
11	T				N	**0.1752**	**0.1449**	**0.1723**	**0.1297**
12	T	FB				0.1296	0.1155	0.1074	0.0986
13	T		FB			0.1331	0.1211	0.1095	0.1022
14	T			FB		**0.1579**	**0.1357**	**0.1240**	**0.1052**
15	TRD	N				0.2678	0.2161	0.2433	0.1998
16	TRD		N			0.2677	0.2159	0.2433	0.1997
17	TRD			N		0.2544	0.1785	0.2279	0.1629
18	TRD				N	**0.2804**	**0.2235**	**0.2560**	**0.2075**
19	TRD	FB				0.2225	0.1728	0.1945	0.1544
20	TRD		FB			0.2239	0.1749	0.1950	0.1563
21	TRD			FB		**0.2300**	**0.1627**	**0.1971**	**0.1449**

Table 2. Results for Average Precision (LARGE)

No.	Query	Method Used				Average Precision			
		Score	Each	Top	SN	qp-cont	qp-link	q-cont	q-link
22	D	N				**0.1278**	0.1269	**0.0927**	**0.0987**
23	D		N			0.1262	0.1241	0.0913	0.0961
24	D			N		0.1075	0.1042	0.0757	0.0777
25	D				N	0.1273	**0.1270**	0.0920	0.0982
26	D	FB				0.0729	0.0809	0.0697	0.0837
27	D		FB			0.0919	0.0984	0.0829	**0.0952**
28	D			FB		**0.1136**	**0.1101**	**0.0899**	0.0922
29	T	N				**0.1386**	**0.1369**	**0.0916**	**0.1007**
30	T		N			0.1365	0.1330	0.0901	0.0972
31	T			N		0.1166	0.1117	0.0719	0.0764
32	T				N	0.1381	0.1364	0.0914	0.0985
33	T	FB				0.0746	0.0859	0.0686	0.0886
34	T		FB			0.0923	0.1022	0.0802	**0.0988**
35	T			FB		**0.1094**	**0.1089**	**0.0823**	0.0888
36	TRD	N				0.1515	0.1461	0.1271	0.1294
37	TRD		N			0.1507	0.1448	0.1263	0.1281
38	TRD			N		0.1232	0.1052	0.1059	0.0911
39	TRD				N	**0.1581**	**0.1510**	**0.1334**	**0.1333**
40	TRD	FB				0.0866	0.0863	0.0774	0.0821
41	TRD		FB			0.0963	**0.0962**	0.0850	**0.0897**
42	TRD			FB		**0.1033**	0.0920	**0.0917**	0.0802

Table 3. Results for R-Precision (SMALL)

No.	Query	Method Used				qp-cont			qp-link		
		Score	Each	Top	SN	P@5	P@10	Ave.	P@5	P@10	Ave.
1	D	N				**0.2174**	0.1717	0.1867	**0.2723**	0.2298	0.1932
2	D		N			**0.2174**	0.1717	0.1867	**0.2723**	0.2298	0.1932
3	D			N		0.1739	0.1565	0.1795	0.2213	0.2043	0.1663
4	D				N	0.2130	**0.1739**	0.1946	0.2681	**0.2383**	0.1951
5	D	FB				0.2130	0.1739	0.1664	0.2426	0.2085	0.1629
6	D		FB			0.2130	0.1739	0.1664	0.2426	0.2085	0.1629
7	D			FB		0.2130	**0.1783**	0.1810	**0.2511**	**0.2170**	0.1836
8	T	N				0.1913	0.1761	0.1757	0.2468	0.2191	0.1782
9	T		N			0.1913	0.1761	0.1757	0.2468	0.2191	0.1782
10	T			N		0.1783	0.1478	0.1731	0.2255	0.1936	0.1591
11	T				N	**0.2000**	**0.1804**	0.1734	0.2468	**0.2234**	0.1851
12	T	FB				0.2000	0.1630	0.1611	0.2340	0.2064	0.1636
13	T		FB			0.2000	0.1630	0.1611	0.2340	0.2064	0.1636
14	T			FB		**0.2043**	**0.1826**	0.1685	0.2340	**0.2149**	0.1751
15	TRD	N				0.3244	0.2489	0.2655	0.4217	0.3326	0.2393
16	TRD		N			0.3244	0.2489	0.2655	0.4217	0.3326	0.2393
17	TRD			N		0.3022	0.2333	0.2477	0.3217	0.2717	0.2005
18	TRD				N	**0.3442**	**0.2628**	0.2820	**0.4409**	**0.3455**	0.2449
19	TRD	FB				**0.2978**	**0.2267**	0.2377	**0.3652**	**0.2891**	0.2050
20	TRD		FB			**0.2978**	**0.2267**	0.2377	**0.3652**	**0.2891**	0.2050
21	TRD			FB		0.2800	0.2089	0.2449	0.2913	0.2348	0.1982

Table 4. Results for R-Precision (LARGE)

No.	Query	Method Used				qp-cont			qp-link		
		Score	Each	Top	SN	P@5	P@10	Ave.	P@5	P@10	Ave.
22	D	N				**0.2638**	0.2660	0.1886	**0.3106**	0.3085	0.1993
23	D		N			**0.2638**	0.2660	0.1888	**0.3106**	0.3085	0.2001
24	D			N		**0.2638**	0.2617	0.1639	**0.3106**	0.3021	0.1718
25	D				N	0.2511	**0.2681**	0.1916	0.3021	**0.3128**	0.1951
26	D	FB				**0.2426**	0.2234	0.1218	**0.2681**	0.2532	0.1341
27	D		FB			**0.2426**	0.2234	0.1306	**0.2681**	0.2532	0.1473
28	D			FB		0.2298	**0.2362**	0.1774	0.2468	**0.2596**	0.1851
29	T	N				**0.3064**	**0.2745**	0.1918	**0.3617**	**0.3191**	0.1998
30	T		N			**0.3064**	**0.2745**	0.1919	**0.3617**	**0.3191**	0.2000
31	T			N		0.2979	0.2489	0.1660	0.3489	0.2915	0.1730
32	T				N	0.2894	0.2574	0.1895	0.3404	0.3064	0.2004
33	T	FB				0.2000	0.2106	0.1225	0.2298	0.2468	0.1394
36	T		FB			0.2000	0.2106	0.1332	0.2298	0.2468	0.1543
37	T			FB		**0.2468**	**0.2404**	0.1734	**0.2638**	**0.2553**	0.1812
38	TRD	N				0.4043	0.3043	0.1879	**0.4913**	0.3870	0.1940
39	TRD		N			0.4043	0.3043	0.1877	**0.4913**	0.3870	0.1942
40	TRD			N		**0.4348**	0.3109	0.1561	0.4609	0.3326	0.1477
41	TRD				N	0.3909	**0.3250**	0.1999	0.4727	**0.4000**	0.2034
42	TRD	FB				0.2478	0.1957	0.1257	0.3000	0.2391	0.1310
42	TRD		FB			0.2478	0.1957	0.1309	0.3000	0.2391	0.1373
43	TRD			FB		**0.2957**	**0.2413**	0.1495	**0.3174**	**0.2609**	0.1432

6 Conclusion and Future Work

We examined four methods of merging multiple results from retrieval systems.

We could find that the combination of the Score method and method N was the most efficient approach. Result were similar to the SN method. Furthermore, we could find that when using method FB, the Top method is the most efficient approach. This finding means that when the Web documents are automatically collected, their contents are uniformly distributed and the OKAPI method is used by the search engine, the score information obtained from the results can be used in Web retrieval without score normalization.

In general, we could not get the score information from each search engine, as they use different methods. Our results confirm that the Top method is the most efficient.

Results for average precision are not satisfactory for an information retrieval system, although the IR package is a satisfactory tool in retrieving news articles. Thus, we need to develop some revised methods of retrieving unarranged documents. In this study, we did not use link information or an interactive mode of the IR package. Previous research [4] has shown the effectiveness of using link information. We would like to investigate the approach of using link information.

Results of average precision for the LARGE collection are worst than for the SMALL collection. It can be considered to affect elimination of documents badly. We should cope with processing incorrectly converted files and text files with long lines. In this study, we eliminated all such files. We would like to develop ways to deal with text files with long lines.

Further research will include selecting relevant documents with Rocchio's feedback by using the interactive mode to achieve more effective results. It should also be possible to improve the precision of information retrieval by applying link information.

In this study, we did not consider the processing time required for information retrieval. Search engines need to respond quickly to users' requests. We should thus make efforts and decrease the processing time, although we think the processing time was able to be shortened considerably by omitted score normalization.

References

[1] ftp://ftp.cs.cornell.edu/pub/smart/trec_eval.v3bgeta.share.

[2] J. A. Aslam and M. Montague. Models for metasearch. In *SIGIR 2001*, 2001.

[3] R. Baseza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[4] N. Craswell, D. Hawking, and S. Robertson. Effective site finding using link anchor information. In *SIGIR 2001*, 2001.

[5] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving hightly relevant documents. In *SIGIR 2001*, 2001.

[6] S.E. Robertson and S. Walker. Okapi/keenbow at trec-8. TREC 8 proceedings.

[7] M. Uehara, N. Sato, T. Yamamoto, Y. Nishida, and H. Mori. Distributed scoring in cooperative search engine, 1999. Internet Conference 1999 (in Japaneses).

[8] M. Utiyama and H. Isahara. Implementation of an IR package. IPSJ SIG notes 2001-FI-63 (in Japaneses).

[9] E. M. Voorhees and D. M. Tice. Building a question answering test collection. In *SIGIR 2001*, 2001.