Fujitsu Laboratories TREC9 Report

Isao Namba

Computer System Laboratory Fujitsu Laboratories Ltd.
{namba}@jp.fujitsu.com

Abstract

This year a Fujitsu Laboratory team participated in web tracks. For TREC9 we experimented passage retrieval which is expected to be effective for Web pages which contain more than one topic. To split document into passages, we used NLP based paragrah detecting program, not by fixed (variable) window size. But it did not produce better result for TREC9 Web data. For indexing large web data faster, we developped two techiniques. One is multi-partional selective sorting for inversion which is about 10-30% faster than normal quick sorting in sorting term-number, text-number pair. The other is compressed trie dictionary based stemming.

1 System Description

Except reranking by passage retrieval, and passage segementing program for index preprocessing, the frame work we used, is same as that of TREC8[1].

1.0.1 Teraß

Teraß[2] is a fulltext search library, designed to provide an adequate number of efficient functions for commercial service, and to provide parameter combination testing and easy extension for experiments in IR. For TREC9 we added functions for run time passage retrieval

1.0.2 trec_exec

trec_exec is designed for automatic processing of TREC. It contains a procedure controller, evaluation module , logging module, and all non-searching units such as query generation, query expansion and so on. trec_exec can execute all the TREC processing for one run in a few minutes, and it can be used for system tuning by hill-climing. But it was difficult to tune parameter control for TREC9 web data, because document set and queries for TREC9 is different from past trec data.

2 Common Processing

2.1 Indexing/Query Processing

2.1.1 indexing vocabulary

The indexing vocabulary consists of character strings made up of letters, numbers, and symbols, and no stop words were used in indexing. For TREC8, we modified the grammar of the token recognizer to accept acronyms with symbols such as U.S., and AT&T as one token.

2.1.2 Stemmer

As the experiment in TREC8[1] shows, SMART[3] stemmer seems to be stable, we used SMART.

2.1.3 Information in inverted file

Text number, term frequency, and term position are stored for the ad hoc task, and small web track for run time phrase processing and reranking by bigram extraction.

For experiment of passage retrieval, the delimiters of passage were also indexed.

2.1.4 Stop word list for query processing

As in the TREC8[1], we used a stop word list of about 400 words of Fox[4], and words with a high df (more than 1/7 of the number of all documents) were also treated as stop words.

2.1.5 Stop pattern removal

The expression of TREC queries are artificial, so frequently appearing patterns such as "relevant document" are stop patterns. We generalized this observation, and removed the words which meet one of the following condition.

1. Word in stopword list is a stopword.
2. Word which is not a proper noun[1], and whose df in TREC1-7 queries is more than 400*0.1 is a stop word.
3. Word bi-gram whose df in TREC1-7 queries is more than 400*0.02 is a stop pattern.
4. Word tri-gram whose df in TREC1-7 queries is more than 400*0.01 is a stop pattern.
5. All the words in a sentence that contains "not relevant" are stop words.
6. 4 words following "other than" are stop words.
7. 4 words following "apart from" are stop words.

2.2 Weighting Scheme

The term weight is $qtf * tf * idf$, and the score for one document is the sum of the term weights with co-occurence boosting.

1. qtf

   qtf is the combination of the following parameters

   $qtf = \sum_f fw * tf * ttw$

   where

   f is the topic field (title, description or narrative).
   fw is weight of the topic field. We set the value for the title field to 3.0, the value for the description field 1.5, the value for the narrative is 0.9. Some teams [5], [6],[7] used weighting depending on field type, and we take the same approach.

   tf is the bare frequency in each field.

   ttw is the term type weight. It is set to 3 for terms, and set to 1 for phrase(word bi-gram).

2. tf

   We simply used the tf part of OKAPI[5].

   $tf = \frac{(k_1+1)*term\_freq}{(k_1((1-b)+\frac{b*doc\_length\_in\_byte}{average\_doc\_length\_in\_byte})}$
   $k_1 = 1.5, b = 0.75$

3. idf

   We used a modified idf of OKAPI. We introduced a cut off point for low df words, and decreased the idf value for high df words.

   $idf = log_2 \frac{N-(n*\alpha)}{n}$
   N is the number of documents
   n is df if (df > 1/10000 * N) else
   n = 1/10000 * N
   α is set to 3

[1]U.S appears 94 times in TREC1-7 queries.

2.3 Co-occurence Boosting

As in TREC8, we use co-occurence boosting techinique which favours co-occurence of query terms in a document. Co-ocurrence boosting is implemented by simply multipling the boost ratio to the similarity of each term.

$$S_i = \sum_t B * W_{t,i}$$

S_i is the degree of similarity between a document and topics.
i is the document number.
t is a term that document$_i$ includes.
$W_{t,i}$ is the part of similarity of term$_t$ in document$_i$.
B is the boost-ratio by term co-occurrence.

The best parameter B depends on the query, but it is difficult to tune them for each query. So we set the B to 1.10 for the title word, to 1.05 for the description word, and to 1.03 for the narrative word, and to 1.0 for the word added by query expansion.

2.4 phrase(bi-gram)

Instead of traditional IR phrase (two adjacent non-stopword pair with order or without order), we permitted limited distance in phrase. The motivation for introducing fixed distance is that that non-stopword may exist between two adjacent words in a query, and it producued slightly better result in the past experiment.[1] The term weight of bi-gram is fixed as 1/3 of a single word, and the distance is set to 4.

2.5 Query Expansion

Query Expansion was used for the ad hoc task, and small web track. The Boughanem formula[5] was used to select terms.

$$TSV = (r/R - \alpha s/S).w^{(1)} \qquad (1)$$

$w^{(1)}$ is modified and more general version of Robertson/Sparck Jones weight.

The α was set 0.001, and k4 was -0.3, k5 was 1, and k6 was 64. The top 20 documents in the pilot search were supposed to be relevant, and the documents ranked from 500 to 1000 were supposed to be non-relevant. The top ranked 40 words which are not included in original query, which are not included in the stopword list of SMART, whose tsv score are more than 0.003, whose df are more than 60, and whose df are less than 200000 were added to the original query.

No collection enrichment technique was used.

2.6 Passage Retrieval

The average text size of TREC8 web data is large compared with past TREC collections. Its average text size is about 8KB. If the large web page contains more than one topic, scoring the page by its contents (large passage) not whole contents may produce better result. This requires techniques to split text by structure of topics. Using NLP techiniques developped for text summarization[8], we splitted the text into paragrahs, and indexed the text with topic boundary.

Following is the example of splitted text. The topic delimiter is <delim > tag, and the attribute "level" expresses level of paragraph. As the number of level becomes bigger, the size of paragrah becomes larger.

<document>
<DOCNO>WTX049-B01-2</DOCNO>
<BD>
<delim level=7/>
Table of Contents First-Time Startup
Overview of the First-Time Startup Process Default Values Using the Setup Command Facility
Help Text Using the Setup Command Facility Powering Up Your System Verifying Installed Software and Hardware
Configuring Global and Interface Parameters Storing the Configuration in Nonvolatile Memory
Sample Configuration
<delim level=1/>
This chapter includes sample worksheets filled in to show you how this information is used when the setup command facility runs through the System Configuration Dialog.
Note Some configuration parameters discussed in this document (and shown on the configuration worksheets) apply
only to routers that have the protocol translation option. If your router does not have protocol translation,
the interactive setup command facility does not prompt you for these parameters.
<delim level=1/>
Overview of the First-Time Startup Process
The first time you start up the system, the setup command facility operates automatically. An interactive
dialog called the System Configuration Dialog appears on the system console screen. The dialog navigates you
through the configuration process by prompting you for the information you have recorded on the configuration
worksheets. The setup command facility also provides default values and help text for the configuration parameters,
as described later in this section.
The setup command facility detects which interfaces are installed and prompts you for configuration information
for each installed interface. When you finish configuring one interface, the setup command software prompts you
for the next interface and continues until they are all configured.
At first-time startup, you must do the following:
Power up your router and if necessary, test for problems with system memory and CPU.
Verify software version and installed hardware and software options.
Configure global parameters.
Configure interface parameters.
<delim level=2/>
:
:
Copyright 1988-1995
Cisco Systems Inc.
</BD>
</document>

We simply apply Okapi scoring (variation we used) to the passage, and merged fulltext scoring, and passage scoring. In the training by TREC8 web data, we set passage boundary level to 3, in that case average passage size was about 250 words. Merging his technique prodocues slightly better (1

point in average precision) result for TREC8 web track data, but did not result in improvment in TREC9 web track data whose average text size is about 4KB

3 Small Web Track official Runs

Four runs are submitted, Flab9atN, Flab9atdN, Flab9atd2N, and Flab9atdnN. In the Run id, the infix 'a ' means automatic, 't' means using title field, 'd' means using description field, and 'n' means using narrative field.

Name	Flab9t	Flab9tdN	Flab9td2N	Flab9tdnN
field	T	TD	TD	TDN
link	NO	NO	NO	NO
Average Prec	.136	.181	.187	.192
R-Prec	.153	.207	.208	.223
P@20	.157	.232	.226	.252
Retrieved	50000	50000	50000	50000
Rel-ret	2617	2617	2617	2617
Relevant	1179	1526	1490	1567
best/ >= med	2/25	0/31	0/31	0/35

Table 1: Official web track result

4 Speed up of indexing

Generally sorting based inversion takes these 4 steps.

1. STEP1 Apply stemmer to input text.
2. STEP2 Convert stemmed word to term-id. In most cases term-id to stemmed word is assigned by sequential order. Using hash may be fastest.
3. STEP3 Put [term-id,text-number,(offset)] pair(tuple) to work area
4. STEP4 If work area is full, then sort the area by ascending order of term-id,text-number,offset.

In sorting based inversion, stemming(+hashing) and sorting takes 70% of whole processing speed[1]. So speed up of above process leads to speed of whole processing.

4.1 Multi-paritional selective sorting

The fastest sorting algorithm is generally quick sort algorithm. But in sorting the pair of inverted file entry, we can expect the distribution of primary key(term_id). Because the word with high document frequency gets the smaller number, and the word with low document frequency gets the bigger number, they distribute in a log regression manner. Using this statistics, we can partion the sorting area into multi blocks at one time instead of partioning the sorting area in binary block (quick sort). Multi block partioning soring is faster than binary partioning sorting if partitioning is successful. [2]

The other techniques we introduced is using radic sort. The order of radic sort is $O(n)$, so it is expected to be faster than quick sort $On(log_2(n))$. But in practice, it is slower than quick sort for large data. It is because radic sort requires two buffers, and once copying between two buffers requires real memory access, it serverly slow down. But if sort target is small enough [3], it is surely faster than quick sort.

Using quick sort for large block, and radic sort for small block, we can improve the sorting speed of overall for inverted file entry.

The multi-paritional selective sorting algorithm is as follows.

1. Input is [term_id,text_number]
2. Prepare $n+1$ blocks. n is $log_2(maxterm\_id)$
3. Parition entry into $n+1$ blocks. The partitioning function is to put entry to $log_2(term_id)$th block.
4. Foreach blocks, apply sorting.
   (a) If partion is larger than 3/4 of L2 cache, use quick sort.
   (b) If partion is small than 3/4 of L2 cache, use radic sort.

The performace of this approach is depending on the partioning. In the experiment for VLC100, it is 10% faster than normal quick sorting, for WT10g, it is 30% faster than normal quick sorting. [4]

[2] multi partioning itself requires more complicated decision function than binary blocking sorting. So if partioning results in unbalanced blocks, it is slower than quick sort.

[3] it depends on the L2 cache of Hardware

[4] The speed depends on the max text_number in sorting target, and target area size etc.

4.2 Speed up of stemmer

As the stemming program matches rules step by step, it is slower than simple token recognizer. For example just recognizing token can process 16.5GB documents per hour, but with SMART stemmer it slows down 6.4GB documents per hour. Making compressed trie dictionary of frequently appearing 70000 words in text, and skip running stemming algorithm for them, the speed of stemmer increases about 13%.

5 Large Web Track

Our main concern this year is still how much resources are required to processing large data.

We concentrated on speed up of indexings.

5.1 Hardware environment

One PC was used for the large web track. It has 120GB disk, 640MB main memory and two Intel Celeron 466MHz CPUs. Its cost is about 180000 Japanse yen (about 1700 US$) at Nov 1999.

Using PC for information retrieval has practical advantages.

One is that PC(Intel i86) is cheaper than workstation. This is well known. The other is that the speed of information retrieval process is depending on the performance of integer calculation.[5] The PC(i86)'s processing speed of floating point calculation is slower than that of workstation, even its clock is 3 times faster than workstation, but this disadvantage is not critical in IR application.

5.2 language type checking

To reduce index size, and increase the speed of searching, statistical based language type checker is used as in TREC8[1]. Its effect is that the sum of word entry in inverted file is reduced to 10 million from 20 million, and the index size is reduced to 4.0GB from 4.8GB without stopword condition.

5.3 Large web track result

Our main concern is balancing processing speed and hardware cost. The submitted 3 runs are the same condition as that of TREC8. All runs did not use phrases, and query expansion. B+R means ranking document with AND condition of every non-stopword in a query. If the number of retrieved documents is less than 20, then ranking search is retried. This AND conditional interface is popular in actual Internet services. R means traditional accumulator method. Flab9bsN used index with stopwords. Table2 shows our official result.

Run-id	P@5	P@10	Calc	speed(sec)
Flab9bN	0.44	0.46	B+R	0.31
Flab9rN	0.44	0.45	R	0.72
FLab9bsN	0.45	0.45	B+R	0.47

Table 2: Large web official result

There is no remarkable difference in precision. B+R search and index with stopwords seems to be the best choice considering speed.

5.4 Performance of pre-processing

Compared with TREC8, we improved preprocessing speed. The preprocessing involves web detagging, running language type checking, and indexing. The official pre-processing data is as follows.

1. The detagging script for TREC8[1] is rewritten in C, and its processing time is improved to 10 hours including the time for gunzip the data.
2. language type checker takes 4 hours using 2 CPU.
3. Indexing

   Instead of Solaris2.6, we used Linux to avoid work memory swapping out problem. [1] The indexing time is 10 hours and 6 minutes with stopword condition.

condition	time	status	work area
With stopword	10.13 hours	official	300MB
Without stopword	12.15 hours	official	300MB

Table 3: Inversion time

The Index size is given in table 4.

[5] Ranking requires floating point calculation, but the most of CPU time is used for logical operation and decoding of inverted file entry.

files	with stopword	without stopword
inverted file	3.01	4.03
dictionary	0.46	0.59
text size array	0.07	0.07
text number id	0.41	0.41
total	3.95GB	5.10GB

Table 4: Index size

5.5 Performance of query processing

5.5.1 Average Processing Speed

The regulation of a large web track says that query processing speed is the total processing time divided by the number of query, As the experiment TREC8[1] shows, using two processs in 2 CPU environments is fastest. In TREC9 we used thread based approach. [6] The query processing speed is given in official result.

6 Conclusion

For small web track, we tried applying passage based scoring, but we did not get improvment. For large web track, we used two techniques, both of which are effective for speed up indexing.

References

[1] I Namba and N Igata. Fujitsu laboratories trec8 report. *The Eighth Text REtrieval Conference*, 2000.

[2] I Namba, N Igata, H Horai, K Nitta, and K Matsui. Fujitsu laboratories trec7 report. *The Seventh Text REtrieval Conference*, 1999.

[3] SMART ftp cite. ftp://ftp.cs.cornell.edu/pub/smart/. 1999.

[4] Chiristopher Fox. Chapter 7, lexical analysis and stoplists. *Information Retrieval Data Structure and Algorithms ed. William B. Frakes, Ricardo Baeza-Yates Prentice Hall*, 1992.

[5] S E Robertson, S Walker, and M Beaulieu. Okapi at trec-7. *The Seventh Text REtrieval Conference*, 1999.

[6] D R H Miller, T Leek, and R M Schwarts. Bbn at trec-7. *The Seventh Text REtrieval Conference*, 1999.

[7] James Allan, Jamie Callan, Mark Sanderson, Jinxi Xu, and Steven Wegmann. Inquery and trec-7. *The Seventh Text REtrieval Conference*, 1999.

[8] Yoshio Nakao. Summary zation by structure of topics(in japanese). *IPSJ Natural Lanuage Processing Group 132th meeting*, 1999.

[6] Using multi-threading techniques, we tried more complicated processing in experiment, but we don't report the experiment in this paper.