Kasetsart University TREC-9 Experiments

P. Narasetsathaporn, A. Rungsawang
{g4365020,fenganr}@ku.ac.th
Massive **I**nformation & **K**nowledge **E**ngineering
Department of Computer Engineering
Faculty of Engineering
Kasetsart University, Bangkok, Thailand.

Abstract

We add pivoted unique normalization weighting scheme to SMART and use it to run the final experiments. Since SMART produces an inverted file which is larger than 2G limitation on our x86 based Linux machine, we have then to divide small web track test set into several subcollections, index and retrieve, and merge all scores to obtain the final result.

1 Introduction

In our TREC-8 experiments last year, we proposed to mine good candidate terms from the document collection, using "Apriori algorithm" [1], and then use that terms to enhance the original query [3]. We then used the new expanded query to retrieve relevant documents in the collection. From those experiments, we had found that the proposed technique worked well with the FT (Financial Time) collection using some weighting pairs, such as lnc.ntc or lnc.atc, and we got till 19% improvement.

In the TREC-9 experiments this year, we try to use the same technique with the whole small web track documents, but confront with many technical obstacles. Firstly, we have spent very much time to write a robust parser to parse messy data in small web track documents. Secondly, the number of different terms, after removing stopwords and stemming, is so numurous that the Apriori algorithm we use to mine good candidate terms, as well as our own DSIR text retrieval algorithm, have hardly come through in time we have left, though using the most powerful x86 PC based machine (equipped with 512M of RAM) we have in our department.

We then decide to modify the Cornell's SMART version 11.0 so that it can run smoothly on our Linux machine, and add the notable pivoted unique normalization weighting scheme [4] to it. However, another intrinsic operating system problem arises. We cannot index the whole small web track documents in one-shot since SMART will generate an inverted-file image that is larger than the 2G limitation of the x86 based Linux machine. Therefore, we have to split the small web track collection into

several sub-collections, index and test those sub-collections with SMART, merge all subsequent results to get the final top-1000 scores.

The rest of our report gives more detail about what we do during TREC-9 period. Sections 2 describes new weighting scheme that we add to the original SMART version 11.0. Sections 3 gives more detail about the merging algorithm we use to combine the whole final scores from several runs. Sections 4 provides the final results we obtain, and section 5 concludes this report.

2 Adding New Weight to SMART

Since we found that retrieval results recieved from the classical weighting schemes provided by the original SMART distribution are not as good as we expect, and unfortunately there is no patch for the new well-known weighting schemes, we then decide to mess up some more codes into the original SMART. We have followed the pivot document length normalization weighting scheme introduced by Singhal et al. [4]. This normalization scheme is based upon both normalizing the t_f (term-frequency) factor by the average t_f in the document vector, and the overall vector length by a pivot and a slope factor dependent on the number of unique terms in that document. Based on the underlying t_f factor (which we call the L factor in the SMART tripple weighting notation), and the pivoted unique normalization (which we call the u normalization), we obtain the final weighting scheme, called Lnu weighting in SMART, in the form of:

$$\frac{\frac{1+\log(t_f)}{1+\log(\text{average } t_f)}}{(1.0-\text{slope})*\text{pivot}+\text{slope}*\#\text{of unique terms}} \quad (1)$$

which the #of unique terms is the amount of term of which t_f is equal to 1, and pivot is the average number of unique terms.

To obtain this weight, we modify the SMART version 11.0 in src/lib-convert by adding the function tfwt_triple (L) in weights_tf.c and norm-wt_unique (u) in weights_norm.c, while during experiments the pivot and slope are read from the spec file. We also add the L and u to tell SMART about this new weight in src/libproc/proc_convert.c

After adding the new weight to SMART, we verify it by running some retrieval experiments. We choose the FR and FBIS, and the topics 401-450 as our test sets. We obtain the results as illustrated in the Table 1 as follows. Results from this test make us quite certain that we do add the correct weight to the old orginal SMART.

Collection	# of relevance	11pt avg precision		Relevant retrieved	
		lnc.ltc	Lnu.ltu	lnc.ltc	Lnu.ltu
FBIS	1667	0.2001	0.2732	1144	1106
FR	206	0.1814	0.2713	170	170

Table 1: Testing results of our modified SMART.

3 Merging Results from Sub-collections

Since the total number of documents in small web track collection, 1,692,096 documents, is quite large for our x86 based Linux machine, the inverted file index produced by SMART becomes messy when it breaks the 2G barrier. We then have to divide the whole collection into several ones, index and retrieve the top-ranked documents, and merge the whole together to get the final ranking scores. There exist several merging algorithms mentioned in the literatures [2, 5]. We investigate and implement four of them, i.e. interleaved merge, raw score merge, normalized score merge, and weighted score merge, and test them using the FR collection. Results from this test shows that the weighted score appraoch gives the best merging results. The weight w, below, is the one we use to combine the whole final ranking scores from several small web track sub-collections in our TREC-9 experiments.

$$w = 1 + |C| * \frac{s - \bar{s}}{\bar{s}} \tag{2}$$

where $|C|$ is the number of sub-collections, s is the collection's score, and $\bar{s}$ is the mean of the collection scores. With this approach, each document is ranked based upon the product of its score and the weight w for its collection [2].

4 Experiments and Results

We first parse all html tags, images, all messy data, and the others, out of the small web track collection. We use every words found in the small web track topics as queries. From several experiments we have performed till the deadline of this final report, we obtain the best final scores when the original small web track collection has been divided into 7 sub-collections, in a round-robin fashion, as concluded in Table 2 as follows.

Sub-collection	Directory	Doc-number
1	WTX001-WTX015	1-251745
2	WTX016-WTX030	251746-485635
3	WTX031-WTX045	485636-730835
4	WTX046-WTX060	730836-976633
5	WTX061-WTX075	976634-1233895
6	WTX076-WTX090	1233896-1474263
7	WTX091-WTX104	1474264-1692096

Table 2: Small web track sub-collections.

We index all the sub-collections seperately, using our own modified SMART running on a x86 based Linux machine, and try with several weighting schemes. We found that the weight Lnu.ltu combination gives the best scores, when pivot and slope parameters in Equation 1 have been set as shown in the Table 3 as follows.

Sub-collection	Pivot	Slope
1	84.53	0.15
2	90.86	0.1
3	83.44	0.05
4	81.05	0.05
5	80.54	0.1
6	82.98	0.25
7	84.15	0.2

Table 3: Pivot and slope parameters.

Using weighted score merging approach, the final result we have is concluded in the Table 4 below. Note that we also give the result obtained from lnc.ltc weight for reference.

Weight	Avg.Precision	Rel.Retrieved	P.@5doc	P.@100doc
lnc.ltc	0.0525	803	0.1080	0.0508
Lnu.ltu	0.1943	1122	0.3360	0.1058

Table 4: KU TREC-9 final small web track results.

5 Conclusion

In our TREC-9 experiment this year, we have not provided any valuable finding to the web track community, but just participate in spirit. Till the last day of this final report deadline, we do hardly come through to get some results. We have spent a lot of time to code a robust parser to extract free text from small web track documents. We lost much time to alter the Apriori algorithm to run with big small web track data, but it has never been converted to give any results. We also face with the intrisic Linux operating system problem when the file size is larger than 2G barrier on the x86 machine.

Since we do not succeed to let the Apriori algorithm convert on the whole small web track data, our claim last year about the query enhancement technique [3] is still not verified. We turn to use our own modified version of SMART to run the experiments. We add the new weight, i.e. pivoted unique normalization, in SMART. Since the problem of 2G file size limitation in our Linux based machine still exists, we then divide the small web track test set into several sub-collections, index and retrieve relevant documents seperately, and use weighted score merging technique to combine the final top-1000 scores.

Ackknowledgement

We would like to thank our MIKE staffs, especailly Hong, Wit, Tew, for their programming support and working spirit. We also thank to Ink who worked so hard with Apriori algorithm last year.

References

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the* 20^{th} *VLDB Conference*, Staiago, Chile, 1994.

[2] J.P. Callan, Z. Lu, and W.B. Croft. Searching Distributed Collections with Inference Networks. In *Proccedings of the* 18^{th} *ACM-SIGIR Conference*. ACM Press, 1995.

[3] A. Rungsawang, A. Tangpong, P. Laohawee, and T. Khampachua. Novel Query Expansion Technique using Apriori Algorithm. In E. Voorhees and D. Harman, editors, *Proceedings of the* 8^{th} *Text REtrieval Conference*. NIST Special Publication 500-246, 1999.

[4] A. Singhal, C. Buckley, and M. Mitra. Pivoted Document Length Normalization. In *Proceedings of the* 19^{th} *ACM-SIGIR Conference*. ACM Press, 1996.

[5] E. Vorhees, N.K. Gupta, and B. Johnson-Laird. The Collection Fusion Problem. In D. Harman, editor, *Proceedings of the* 3^{th} *Text REtrieval Conference*. NIST Special Publication 500-225, 1995.