CAS-ICT at TREC 2005 Robust Track: Using Query Expansion and RankFusion to Improve Effectiveness and Robustness of Ad Hoc Information Retrieval

Guodong Ding, Bin Wang and Shuo Bai

Software Division, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China

dingguodong@software.ict.ac.cn,wangbin@ict.ac.cn, sbai@sse.com.cn

ABSTRACT: Our participation in this year's robust track aims to: (1) test how to improve the effectiveness of IR (according to MAP) using different retrieval methods with different local analysis-based query expansion methods; (2) test how to improve the retrieval robustness (according to gMAP) using RankFusion, a novel fusion technique proposed in our experiments. Our results show that although query expansion can improve the effectiveness of IR significantly, it hurts the robustness of IR seriously. However, with appropriate parameters setting, using RankFusion for merging multiple retrieval results can improve the robustness significantly while not harming the average precision too much or even increasing it in some cases.

1 Introduction

The robust retrieval track is a traditional ad hoc retrieval task where the evaluation methodology emphasizes a system's least effective topics. The goal of the track is to improve the consistency of retrieval technology by focusing on poorly performing topics.

This year is the first time for LCC (Large-scale Content Computing) group in ICT to participate in the robust track. The last two years since 2003 showed that the most promising approach to improving poorly performing topics is to utilize external text collections such as the web other than the target collection [1]. But we didn't exploit web-assisted tools such as Google for query expansion, which has been viewed as a very effective strategy for improving retrieval performances [1][2][3]. We argue that Google can be only viewed as web-assisted tool rather than an external web collection, because we don't know any mechanism or retrieval procedure behind Google and what we can do is just to use its retrieval results for enhancing the original query.

In the robust track, we focus on how to effectually improve the retrieval performances (effectiveness and robustness measured by MAP and gMAP[1] [1], respectively) by merging multiple retrieval ranks produced by multiple retrieval methods with or not with query expansion. To rank the documents, we utilize five document scoring functions, called OKAPI-BM25, LMIR-JM, LMIR-DIR, LMIR-ABS, LMIR-GJM2, respectively. To expand the original query, we use three local-analysis based expansion methods, named as LOCOOC, LOCFB, and KLD,

[1] In our experiments, the computation of gMAP is based on [1]. Suppose the number of topics is N, and the average precision of topic *i* is *AvgPrec*(*i*), then:

$$gMAP = \exp\left(\frac{1}{N}\sum_{i=1}^{N}\log\left(AvgPrec(i)+0.00001\right)\right)-0.00001$$

respectively. A novel fusion method ---- RankFusion, which is independent of the retrieval methods and document score, is proposed for merge multiple runs.

The remainder of our paper is organized as follows. In Section 2 we present the basic retrieval architecture of our IR system and summary the five retrieval methods and the three query expansion methods used in the track. In Section 3 we give a formal description of RankFusion. In Section 4 we introduce our query difficulty prediction method. Official submissions and results are presented in Section 5. Conclusions of our work are summarized in Section 6.

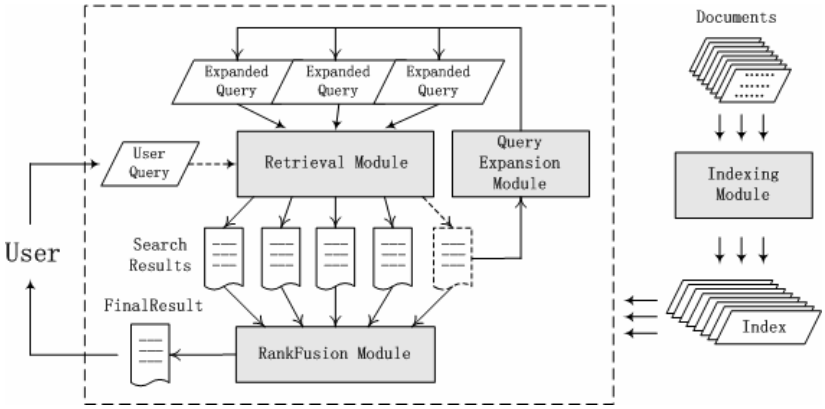

Fig.1 Retrieval architecture of our IR system

2 Retrieval Methods and Query Expansion Methods

The architecture of our IR system is illustrated in Fig. 1, where retrieval module, query expansion module and RankFusion module are the main parts of the system. In retrieval module, we exploit five retrieval methods (OKAPI-BM25, LMIR-JM, LMIR-DIR, LMIR-ABS, LMIR-GJM2), according to the following uniform document scoring form:

$$Sim(D,Q) = \sum_{t \in Q} W(t \mid Q) \cdot W(t \mid D) \tag{1}$$

where $Sim(D,Q)$ is the relevance score of document D given query Q, $W(t|Q)$ and $W(t|D)$ are the weights of term t in Q and in D, respectively.

In the five retrieval methods, OKAPI-BM25 originates from the popular 2-possion based probability retrieval model [4]. The other four methods are derived from the language modeling approach to IR [5][6], each using a different smoothing method: JM denotes Jelinek-Mercer, DIR denotes Dirichlet Priors, ABS denotes absolute discounting, and GJM2 denotes GJM-2 smoothing method [6]. The five retrieval methods are summarized in Table 1 in terms of $W(t|Q)$ and $W(t|D)$ in the uniform form. Table 1 also gives the default values of parameters in each retrieval method.

In query expansion module, we exploit three local-analysis based expansion methods, which use a local document set (generally composed of the top-ranked n documents retrieved by original query) to extract most appropriate terms to expand the query. The core of query expansion is the term ranking function $Score(t,Q|C,S)$[7], which indicates the correlation degree of term t with original query Q given the document collection C and the local document set S. According to $Score(t,Q|C,S)$, the top-scored k terms are added to the query. Table 2 summarizes the three expansion methods, named as LOCOOC[7], LOCFB[7], KLD[8], in terms of $Score(t,Q|C,S)$.

For reweighting the terms in the expanded query Q_{exp}, we use the following function [7]:

$$W(t \mid Q_{exp}) = \alpha \cdot W(t \mid Q) + \beta \cdot \frac{Score(t)}{MaxScore} \quad (2)$$

where $W(t|Q)$ is the weight of term t in original query Q, $Score(t)$ is equivalent to $Score(t,Q|C,S)$, and $MaxScore$ is the maximum score of all the selected expansion terms. The coefficient α and β are both set to 1.0 in all experiments.

Table 1. Summary of the five retrieval methods used for the robust track. $tf(t|Q)$ and $tf(t|D)$ are the frequencies of term t occurring in query Q and document D, respectively. $dl(D)$ is the length of document D, $|D| = dl(D)$. $avdl(C)$ is the average length of all the documents in collection C. $p_{ML}(t|D)$ and $p_{ML}(t|C)$ are the maximum likelihood estimators of the probabilities of term t occurring in document D and collection C, respectively. $|D|_u$ is the number of unique terms in document D.

Retrieval Method	$W(t\|Q)$	$W(t\|D)$	Param Values
rm1: OKAPI-BM25	$\frac{(k_3+1)\cdot tf(t\mid Q)}{k_3+tf(t\mid Q)}$	$\frac{(k_1+1)\cdot tf(t\mid D)}{k_1\left[(1-b)+b\cdot\frac{dl(D)}{avdl(C)}\right]+tf(t\mid D)}\cdot\log\frac{N-df(t\mid C)+0.5}{df(t\mid C)+0.5}$	k_1 = 1.2 k_3 = 1000 b = 0.75
rm2: LMIR-JM	$tf(w\|Q)$	$\log(\lambda p_{ML}(t\mid D)+(1-\lambda)p_{ML}(t\mid C))$	λ = 0.6 (Title),or λ = 0.4 (Desc)
rm3: LMIR-DIR	$tf(w\|Q)$	$\log\left(\frac{tf(t\mid D)+\mu p_{ML}(w\mid C)}{\mid D\mid+\mu}\right)$	μ = 1000
rm4: LMIR-ABS	$tf(w\|Q)$	$\log\left(\frac{\max(tf(t\mid D)-\delta,0)}{\mid D\mid}+\frac{\delta\mid D\mid_u}{\mid D\mid}p_{ML}(t\mid C)\right)$	δ = 0.8
rm5: LMIR-GJM2	$tf(w\|Q)$	$\log\left(\frac{\mid D\mid_u}{\mid D\mid_u+\mu}p_{ML}(w\mid D)+(1-\frac{\mid D\mid_u}{\mid D\mid_u+\mu})p_{ML}(w\mid C)\right)$	μ = 1000

Table 2. Summary of the three query expansion methods used for the robust track. $idf(\cdot|C)$ is inverse document frequency of a term in the collection. $cood(t,q|S)$ is the co-occurrence degree of the term t and the query term q in S, see[7]. $W(t|D)$ is the weight of term t in document D, generally using the weighting form in OKAPI-BM25. $p(t|S)$ and $p(t|C)$ are the probabilities of term t occurring in S and in collection C, respectively.

QE Method	$Score(t,Q\|C,S)$
qe1: LOCOOC	$\sum_{q\in Q} idf(q\mid C)idf(t\mid C)\log(cood(t,q\mid S)+1.0)$
qe2: LOCFB	$\frac{1}{n}\sum_{D\in S}W(t\mid D)$
qe3: KLD	$p(t\mid S)\cdot\log\frac{p(t\mid S)}{p(t\mid C)}$

Table 3 gives the retrieval performances of the three expansion methods and their improvements over the initial retrieval results (i.e. no expansion is used) for each retrieval method. From the table we can see that in all retrieval methods, although query expansion can improve average precision (MAP) significantly, it can usually hurt retrieval robustness (measured by gMAP) seriously. These observations are also testified in previous robust tracks [1].

Table 3. The retrieval performances (MAP and gMAP) of the three expansion methods and their improvements over the initial retrieval results (i.e. no expansion is used) for each retrieval method. The document collection is Disk4&Disk5-CR, with queries constructed only using the title of topics exploited in this year's robust track. Stop-words are removed and Porter-stemming is applied. The "*rmX*" column is ID of retrieval method *X*(see Table 1). "*rmX.qeY(10,80).run*" denotes the retrieval result using retrieval method "*rmX*" and query expansion method "*qeY*" (see Table 2, "*qe0*" denotes no expansion). The expansion parameters are set to (10,80) for all expansion methods, where 10 is the number of top-retrieval documents and 80 is the number of expansion terms.

rmX	*rmX.qe0.run*		*rmX.qe1(10,80).run*		*rmX.qe2(10,80).run*		*rmX.qe3(10,80).run*	
	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP
rm1	0.0915	0.0550	0.1352 +47.8%	0.0442 -19.6%	0.1342 +46.7%	0.0485 -11.8%	0.1331 +45.5%	0.0571 +3.8%
rm2	0.0864	0.0489	0.1325 +53.4%	0.0507 -3.7%	0.1382 +60%	0.0639 +30.6%	0.1350 +56.2%	0.0690 +41.1%
rm3	0.0998	0.0590	0.1187 +18.9%	0.0409 -30.7%	0.1288 +29.1%	0.0537 -9.0%	0.1210 +21.2%	0.0548 -7.0%
rm4	0.0952	0.0548	0.1260 +32.5%	0.0388 -29.2%	0.1315 +38.1%	0.0497 -9.3%	0.1255 +31.8%	0.0509 -7.1%
rm5	0.1027	0.0618	0.1225 +19.3%	0.0421 -31.9%	0.1271 +23.8%	0.0471 -23.8%	0.1211 +17.9%	0.0558 -9.7%

Table 4. The influences of RankFusion to MAP and gMAP. The data set is the same as that used in Table 3. The "*rmX*" column is ID of retrieval method *X*(see Table 1). The "*rmX.qe0.run*" column shows the performances for retrieval method "*rmX*" with no query expansion. Other columns which contain "*rf*" show the performances after using RankFusion for combination: *rmX.qeYrf.run* = 0.2**rmX.qe0.run* ⊕0.8*(*rmX.qeY(10,80).run*⊕*rmX.qeY(30,80).run*), *rmX.qerf.run* = *rmX.qe1rf.run*⊕*rmX.qe2rf.run*⊕ *rmX.qe3rf.run* The last row shows the performances of final integrative results merged by the results produced for each retrieval method.

rmX	*rmX.qe0.run*		*rmX.qe1rf.run*		*rmX.qe2rf.run*		*rmX.qe3rf.run*		*rmX.qerf.run*	
	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP
rm1	0.0915	0.0550	0.1316 +43.8%	0.0598 +8.7%	0.1308 +43%	0.0602 +9.5%	0.1248 +36.4%	0.0639 +16.2%	0.1339 +46.3%	0.0644 +17.1%
rm2	0.0864	0.0489	0.1366 +58.1%	0.0625 +27.8%	0.1364 57.9%	0.0645 +31.9%	0.1304 +50.9%	0.0684 +40%	0.1391 +61%	0.0671 +37.2%
rm3	0.0998	0.0590	0.1249 +25.2%	0.0590 +0%	0.1308 +31.1%	0.0652 +10.5%	0.1212 +21.4%	0.0622 +5.4%	0.1303 +30.6%	0.0654 +10.8%
rm4	0.0952	0.0548	0.1308 +37.4%	0.0571 +4.2%	0.1339 +40.7%	0.0599 +9.3%	0.1220 +28.2%	0.0585 +6.8%	0.1325 +39.2%	0.0598 +9.2%
rm5	0.1027	0.0618	0.1302 +26.8%	0.0621 +0.5%	0.1300 +26.6%	0.0618 +0%	0.1235 +20.3%	0.0645 +4.4%	0.1297 +26.3%	0.0641 +3.7%
qerf.run (*rm1.qerf.run*⊕*rm2.qerf.run*⊕…⊕*rm5.qerf.run*): MAP=0.1391, gMAP=0.0730										

3 Merging Multiple Results with RankFusion

Previous studies indicate that retrieval performances can often be improved by using a number of different retrieval algorithms and combining the results, in contrast to using just a single retrieval

algorithm [8][9][10]. This is because different retrieval algorithms, or retrieval experts, often emphasize different document and query features when determining relevance and therefore retrieve different sets of documents [9].

Many combination methods took a linear combination of the relevance scores, based on Saracevic and Kantor's early observation that the more runs a document is retrieved by the more likely it is that the document is relevant [12]. However the relevance scores may be not comparable across the runs. Some other combination methods tried normalization of relevance scores prior to combination and got very little success, probably due to the different distribution of the scores in each run [8].

In the robust track, the method proposed for results combination is named as RankFusion, which computes the rank score for each retrieved document based on the document's rank order in its result list, and then uses the rank score to reorder all the documents in the combined retrieved document set. Formally, suppose we have m rank lists of retrieved documents for query Q: $\{L_1(Q),\ldots,L_i(Q),\ldots,L_m(Q)\}$. Each rank list $L_i(Q)$ has n_i retrieved documents (n_i is usually equal to 1000 in TREC-style result):

$$L_i(Q) \equiv ((D_{i,1}, s_{i,1}), (D_{i,2}, s_{i,2}), ..., (D_{i,j}, s_{i,j}), ..., (D_{i,n_i}, s_{i,n_i})),\ 1 \le i \le m,\ 1 \le j \le n_i$$

where $s_{i,j}$ is the relevance score of document $D_{i,j}$ in $L_i(Q)$, and $s_{i,j} > s_{i,j+1}$. For the query Q, suppose the final combined rank list $L(Q)$ is

$$L(Q) \equiv ((D_1, s_1), (D_2, s_2), ..., (D_j, s_j), ..., (D_n, s_n))$$

where each document D_j occurs at least one list in the m rank lists and s_j is the relevance score of document D_j. How do we merge the m rank lists $\{L_1(Q),\ldots,L_i(Q),\ldots,L_m(Q)\}$ into $L(Q)$?

In our combination method, we assign a rank score $RankScore(D|L_i(Q))$ to each document D in the rank list $L_i(Q)$. The rank score is computed based on the reciprocal of the rank order[2] $RankOrder(D|L_i(Q))$. That is,

$$RankScore(D \mid L_i(Q)) = 1/RankOrder(D \mid L_i(Q)) \tag{3}$$

For any document that is not in the list $L_i(Q)$, $RankOrder(D|L_i(Q))$ is (n_i+1).

To get the final combined rank list $L(Q)$, we first use the following formula to compute the combined rank score $ComRankScore(D|Q)$ of any document D that occurs at least one list in the m rank lists:

$$ComRankScore(D \mid Q) = \sum_{i=1}^{m} \alpha_i \cdot RankScore(D \mid L_i(Q)) \tag{4}$$

where α_i is a positive coefficient to control the confidence in each rank list.

The $ComRankScore(D|Q)$ reflects integrative estimate of the document D being relevant to the query and can be directly used for measuring the relevance score of D in the final combined list. We reorder all the unique documents in the m rank list according to $ComRankScore(D|Q)$ and return the top-ordered n documents to the user, where the relevance score s_j of document D_j in the final combined list is equal to $ComRankScore(D_j|Q)$. For the sake of simplicity, we introduce an operator "$\oplus$" to denote the combination of multiple results (or runs) in the remainder of the paper:

[2] The rank order of the ith retrieved document in the result list is i.

$$Run = \bigoplus_{i=1}^{m} \alpha_i Run_i = \alpha_1 Run_1 \oplus \alpha_2 Run_2 \oplus ... \oplus \alpha_m Run_m \quad (5)$$

where each combination coefficient α_i is set empirically in our experiments.

In our experiments we exploited various fusion strategies with RankFusion and found that in most cases RankFusion can improve gMAP significantly while not hurting MAP too much. Our experiments showed that for each retrieval method, with appropriate setting of the combination coefficients, using RankFusion to merge the initial retrieval result with the result after query expansion can improve the robustness significantly though it sometimes has some harm for the average precision when compared with query expansion. In addition, merging the results produced by different retrieval methods can also achieve better gMAP, as indicated in Table 4.

4 Query Difficulty Prediction with KL-divergence

Our prediction of query-difficulty is based on the assumption that if there is a significant divergence of query-term distribution in the top-ranked documents and in the total document collection, then we make the hypothesis that this divergence is caused by the query which is easy-defined. That is to say, we can use the negative KL-divergence to predict the query difficulty.

$$DifficultyScore = -\sum_{w \in Q} p(w|S) \log \frac{p(w|S)}{p(w|C)} \quad (6)$$

where $p(w|S)$ is the probability of term w in the document set S composed of top-ranked documents:

$$p(w|S) = \frac{\sum_{D \in S} tf(w|D)}{\sum_{D \in S} \sum_{w \in D} tf(w|D)} \quad (7)$$

and $p(w|C)$ is the probability of term w in the document collection:

$$p(w|C) = \frac{\sum_{D \in C} tf(w|D)}{\sum_{D \in C} \sum_{w \in D} tf(w|D)} \quad (8)$$

5 Official Submissions and Results

In our experiments, according to whether Port-Stemming is used or not, two kinds of indexes are built: STEMMED and UNSTEMMED. We think that using RankFusion to merge the results on STEMMED data set with the results on UNSTEMMED data set other than on a single data set, the retrieval robustness can be improved further. In the robust track, we exploited five retrieval methods (Table 1), with the retrieval parameters being set to default values.

For each retrieval method *rmX* (*X* denotes the id of the retrieval method, see Table 1), we used the three query expansion methods (Table 2) with appropriate parameter settings to obtain multiple retrieval resultsd. For each query expansion method, we experimented with various setting of expansion parameters, primarily including n and k, where n is the number of top retrieved documents and k is the number of expansion terms. We found that in most cases the appropriate values of (n,k) are (10,80) or (30,80) on STEMMED data set. On UNSTEMMED data set, the good setting of (n,k) is (20,60) or (30,60).

For each retrieval method *rmX*, suppose "*rmX.qe0.run"* is the run ID of the retrieval result with the original query (i.e. no query expansion is applied), "*rmX.qeY*(*n*,*k*).*run"* is the run ID of the retrieval result using query expansion method *Y* (see Table 2), with expansion parameters being (*n*,*k*). As [1] mentioned, collection enrichment is a good strategy to improve the retrieval performances of difficult topics. We used GOV data set as an alternative external collection for collection enrichment. First we utilized OKAPI-BM25 to retrieve the GOV collection and extracted 50 terms from the top-ranked 10 documents to form an expanded query. Then for each retrieval method *rmX*, a new result "*rmX.qe-gov.run*" was retrieved using the expanded query. Thus we had total 8 temporary runs for each retrieval method *rmX*, on STEMMED data set and UNSTEMMED data set, respectively. The temporary runs on STEMMED data set are:

rmX.qe0.run, rmX.qe-gov.run, rmX.qe1(10,80).run, rmX.qe1(30,80).run,
rmX.qe2(10,80).run, rmX.qe2(30,80).run, rmX.qe3(10,80).run, rmX.qe3(30,80).run

Runs on UNSTEMMED data set are very similar to them, except the QE parameter (*n*,*k*) is set to (20,60) and (30,60). Using these temporary runs for RankFusion, we submitted 2 title-only (*ICT05qerfT.run*, *ICT05qerfTg.run*) and 2 description-only (*ICT05qerfD.run*, *ICT05qerfDg.run*) runs. The runs whose names contain the label "*g*" are those that use external collection in a first-pass retrieval. Table 5 shows the performances of the final submitted runs. Appendix 1 gives an exhaustive description of the temporary merged runs and their retrieval performances for title-only task.

Table 5. Retrieval Performances of the submitted runs

Run id	Comment	MAP	gMAP	R-Prec	Recall
ICT05qerfTg	Title-only, collection enrichment with Gov	0.2707	0.1888	0.3168	5078/6561
ICT05qerfT	Title-only, no collection enrichment	0.2856	0.1789	0.3259	4967/6561
ICT05qerfDg	Description-only, collection enrichment with Gov	0.2386	0.1425	0.2876	4692/6561
ICT05qerfD	Description -only, no collection enrichment	0.2594	0.1553	0.3053	4807/6561

6 Conclusions

This is the first time for LCC (Large-scale Content Computing) group in ICT to participate in the robust track. We focus on using different retrieval methods and query expansion methods for improving the retrieval effectiveness. Our experiments show that query expansion can hurt robustness seriously while it improves the average precision. We propose a novel combination method – RankFusion, for merging multiple retrieval results. The experimental results show that with appropriate parameters setting, using RankFusion for merging runs can improve the robustness significantly while not harming the average precision too much or even increasing the average precision in some cases.

Acknowledgments

The index of document collection was built using Lemur[3] (version: 3.0) toolkit. We also used some API functions of Lemur for index accessing.

[3] Lemur is a great platform for IR experiments. See http://www.lemurproject.org for more details.

Appendices

1. Performances of temporary and submitted runs for title-only task on STEMMED and UNSTEMMED indexes. The above half part shows temporary results on STEMMED index, while the below half part shows temporary results on UNSTEMMED index. The last two rows give the performances of the submitted runs for title-only. *rmX* denotes the retrieval method with id being "*X*"(see Table 1). *qeY* denotes the query expansion method with id being "*Y*"(see Table 2).

Performanes on STEMMED data set: AQUAINT, Title-only														
rmX	*rmX.qe0.run*		*rmX.qe1rf.run*		*rmX.qe2rf.run*		*rmX.qe3rf.run*		*rmX.qe-gov.run*		*rmX.qerf.run*		*rmX.qerf-g.run*	
	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP
rm1	0.1828	0.1123	0.2778	0.1614	0.2625	0.1556	0.2591	0.1533	0.2146	0.1022	0.2679	0.1612	0.2601	0.1816
rm2	0.1588	0.0958	0.2540	0.1456	0.2334	0.1427	0.2362	0.1375	0.2065	0.0951	0.2495	0.1506	0.2471	0.1756
rm3	0.1991	0.1350	0.2828	0.1649	0.2769	0.1593	0.2775	0.1645	0.2087	0.1003	0.2820	0.1735	0.2691	0.1858
rm4	0.1894	0.1161	0.2844	0.1619	0.2714	0.1525	0.2673	0.1591	0.2140	0.1062	0.2795	0.1592	0.2692	0.1860
rm5	0.1995	0.1321	0.2791	0.1537	0.2713	0.1518	0.2699	0.1608	0.2094	0.1039	0.2780	0.1639	0.2648	0.1813
Final result on STEMMED dataset with no collection enrichment (*qerf.run*): MAP = 0.2876, gMAP = 0.1777														
Final result on STEMMED dataset with collection enrichment (*qerf-g.run*): MAP = 0.2746, gMAP = 0.1934														
Performanes on UNSTEMMED data set: AQUAINT, Title-only														
rmX	*rmX.nqe.run*		*rmX.qe1rf.run*		*rmX.qe2rf.run*		*rmX.qe3rf.run*		*rmX.qe-gov.run*		*rmX.qerf.run*		*rmX.qerf-g.run*	
	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP	MAP	gMAP
rm1	0.1665	0.0913	0.2610	0.1253	0.2518	0.1152	0.2436	0.1150	0.2146	0.1022	0.2532	0.1187	0.2574	0.1604
rm2	0.1501	0.0801	0.2481	0.1245	0.2386	0.1175	0.2313	0.1177	0.2065	0.0951	0.2390	0.1194	0.2470	0.1591
rm3	0.1758	0.0998	0.2488	0.1357	0.2495	0.1399	0.2377	0.1328	0.2087	0.1003	0.2494	0.1452	0.2529	0.1649
rm4	0.1710	0.0926	0.2614	0.1274	0.2594	0.1220	0.2504	0.1261	0.2140	0.1062	0.1290	0.1258	0.2621	0.1587
rm5	0.1746	0.0974	0.2497	0.1314	0.2476	0.1325	0.2371	0.1299	0.2094	0.1039	0.2472	0.1364	0.2514	0.1627
Final result on UNSTEMMED dataset with no collection enrichment(*qerf.run*): MAP = 0.2581, gMAP = 0.1484														
Final result on UNSTEMMED dataset with collection enrichment(*qerf-g.run*): MAP = 0.2620, gMAP = 0.1755														
Performances of submitted runs on Title-only														
ICT05qerfT	**MAP = 0.2856, gMAP = 0.1789**													
ICT05qerfTg	**MAP = 0.2707, gMAP = 0.1888**													

NOTE: The runs on STEMMED data set whose name contains "*rf*" are merged using RankFusion, according to the following fusion strategy:

rmX.qeYrf.run = 0.2**rmX.qe0.run*⊕0.8*(*rmX.qeY(10,80).run*⊕*rmX.qeY(30,80).run*)

rmX.qerf.run = *rmX.qe1rf.run*⊕*rmX.qe2rf.run*⊕*rmX.qe3rf.run*

rmX.qerf-g.run = 0.6**rmXqerf.run*⊕0.4**rmX.qe-gov.run*

qerf.run = *rm1.qerf.run*⊕*rm2.qerf.run*⊕⋯⊕*rm5.qerf.run*

qerf-g.run = *rm1.qerf-g.run*⊕*rm2.qerf-g.run*⊕⋯⊕*rm5.qerf-g.run*

The runs on UNSTEMMED data set whose name contains "*rf*" are merged with similar strategy, except the query expansion parameters were set to (20,60) and (30,60).

The final submitted runs (*ICT05qerfT* and *ICT05qerfT*) were obtained using the following fusion formula:

ICT05qerfT = *stem.qerf.run*⊕*unstem.qerf.run*

ICT05qerfTg = *stem.qerf-g.run*⊕*unstem.qerf-g.run*

where *stem.qerf.run* (or *stem.qerf-g.run*) is the *qerf.run* (or *qerf-g.run*) on STEMMED data set, *unstem.qerf.run* (or *unstem.qerf-g.run*) is the *qerf.run* (or *qerf-g.run*) on UNSTEMMED data set.

References

[1] Ellen M. Voorhees. Overview of the TREC 2004 Robust Retrieval Track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*. 2004.

[2] K.L. Kwok, L. Grunfeld, H.L. Sun, and P. Deng. TREC2004 robust track experiments using PIRCS. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, 2005.

[3] Giambattista Amati, Claudio Carpineto, and Giovanni Romano. Fondazione Ugo Bordoni at TREC 2004. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, 2005.

[4] Robertson, S. E., Walker, S., Jones, G. J. F., Hancock-Beaulieu, and Gatford, M. 1995. Okapi at TREC-3. In *Proceedings of the Third Text Retrieval Conference (TREC-3) (Gaithersberg, Md.)*, NIST Special Publication 500-226, 109-126.

[5] C. Zhai, J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 2004, 22(2): 179~214.

[6] Guodong Ding, Bin Wang. GJM-2: A Special Case of General Jelinek-Mercer Smoothing Method for Language Modeling Approach to Ad Hoc IR. In *Proceedings of the Second Asia Information Retrieval Symposium (AIRS2005)*. 2005. To appear.

[7] Guodong Ding, Shuo Bai, Bin Wang. Local Co-occurrence based Query Expansion for Information Retrieval. In *Proceedings of the 2nd National Conference on Information Retrieval and Content Security (NCIRCS'2005)*, Beijing. 2005. To appear.

[8] Claudio Carpineto, Giovanni Romano and Vittorio Giannini. 2002. Improving Retrieval Feedback with Multiple Term-Ranking Function Combination. *ACM Transactions on Information Systems*, Vol. 20, No. 3: pages 259-290.

[9] Brian T. Bartell, Garrison W. Cottrell and Richard K. Belew. 1994. Automatic combination of multiple ranked retrieval systems. In *proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR'94)(Dublin)*, 173-181.

[10] Weiguo Fan, Michael Gordon, and Praveen Pathak. On linear mixture of expert approaches to information retrieval. *Decision Support Systems*, 2004.

[11] Nick Craswell, David Hawking, Paul Thistlewaite. Merging Results From Isolated Search Engines. *Australasian Database Conference*, 1999.

[12] Saracevic, T. and Kantor, P. 1988. A study of information seeking and retrieving. III. Searchers, seaches, and overlap. *J. Am. Soc. Inf. Sci*. 39, 3, 197-216.