Learning to Rank Query Reformulations

Van Dang, Michael Bendersky and W. Bruce Croft
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{vdang, bemike, croft}@cs.umass.edu

ABSTRACT

Query reformulation techniques based on query logs have recently proven to be effective for web queries. However, when initial queries have reasonably good quality, these techniques are often not reliable enough to identify the helpful reformulations among the suggested queries. In this paper, we show that we can use as few as two features to rerank a list of reformulated queries, or expanded queries to be specific, generated by a log-based query reformulation technique. Our results across five TREC collections suggest that there are consistently more useful reformulations in the first *two* positions in the new ranked list than there were initially, which leads to statistically significant improvements in retrieval effectiveness.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query Formulation

General Terms

Algorithms, Measurement, Performance, Experimentation.

Keywords

Query reformulation, query expansion, query log, query performance predictor, learning to rank.

1. INTRODUCTION

Query logs have become an important resource for many tasks including query reformulation [3, 6]. Most log-based reformulation techniques, however, are evaluated using non-standard approaches and proprietary query logs, making it hard to compare one to another. A more recent study [2] compares different techniques using TREC collections and finds that when intial queries have relatively high quality, query expansion is much more reliable than substitution.

Although the log-based expansion technique [2] can generate some good reformulations for high-quality TREC queries, it also produces many bad reformulations and it does not generate a reliable ranking of the reformulations by quality.

In this paper, we show that we can effectively rerank the list of reformulated queries obtained with this log-based expansion approach. By using as few as two features, *SCQ* (Similarity Collection Query) [8] and *query clarity* [1], we can substantially improve the ranking of reformulated queries in terms of the quality of the reformulations in the top two ranks (measured by *NDCG@2*), which then leads to significant improvements in retrieval effectiveness.

SIGIR'10, July 19–23, 2010, Geneva, Switzerland.
ACM 978-1-60558-896-4/10/07.

2. METHOD

2.1 Log-based Query Expansion

The log-based query expansion method [2] (referred to as `LQE`) is a slight modification of the query substitution method proposed by Wang and Zhai [6]. It first estimates a context distribution for terms occuring in a query log. It then constructs a translation model that can suggest similar words based on their distributional similarity. Given any query, the expansion model will try to expand it with candidates suggested by the translation model for each query term. The model decides whether to expand the query based on how similar the candidate is to the query term and how appropriate it is to the context of the query. For more details, see [2].

2.2 The Reranking Approach

Query quality predictors aim to predict a query's quality without explicit relevance judgements. Thus, given a ranked list of reformulated queries, it is intuitive to think about reorganizing this list based on the "quality" score given by some predictor.

We tried some of the top-performing predictors that Kumaran and Carvalho [4] used in a similar task and found that *SCQ* [8] and *clarity score* [1] are the most effective for our problem. Therefore, we rerank the list of expanded queries by

$$score(q) = \lambda_1 \times SCQ(q) + \lambda_2 \times clarity(q)$$

where λ_1 and λ_2 are weight of the two predictors.

Table 1: Statistics of queries used for reformulation

	AP	WSJ	Robust-04	WT10G	Gov-2
Title Q.	133	133	200	66	119
Desc. Q.	150	150	246	94	134

Table 3: **Evaluation of retrieval effectiveness in terms of *MAP*. * and † indicate significant difference to the original query and LQE's ranked list respectively. Best result in each column is marked in bold.**

	Title Query					Description Query				
	AP	WSJ	RBT-04	WT10G	Gov-2	AP	WSJ	RBT-04	WT10G	Gov-2
Orig-Q	0.1694	0.2594	0.2247	0.1904	0.2829	0.1660	0.2358	0.2519	0.1770	0.2518
LQE	0.1741	0.2563	0.2297	0.1911	0.2559*	0.1694*	**0.2391**	0.2538	0.1775	0.2497
Rerank	**0.1749***	**0.2663***†	**0.2382***†	**0.1962***	**0.2901***†	**0.1820***†	0.2374	**0.2584***†	**0.1836***	**0.2579***†

Table 2: Our approach ("Rerank") consistently outperforms LQE in *NDCG@2*. All differences are significant at $p < 0.05$

Collection	Title Query		Desc. Query	
	LQE	Rerank	LQE	Rerank
AP	0.2434	0.4805	0.2307	0.3728
WSJ	0.2318	0.5040	0.2250	0.3296
Robust-04	0.2905	0.5559	0.2138	0.3687
WT10G	0.2673	0.5499	0.1680	0.3847
Gov-2	0.1933	0.5830	0.2059	0.4093

3. EVALUATION

3.1 Experiment Settings

In this section, we evaluate the performance of our reranking technique. Evaluation is done on five TREC collections: AP, WSJ, Robust-04, WT10G and Gov-2, with both *title* and *description* queries. We use the language modeling framework and remove all stop words at indexing time. We adopt the parameter settings for LQE from the authors [2].

Due to the limited coverage of the available query log [5], we use only a subset of TREC queries where the LQE can generate at least one reformulation. Information about these subsets is given in Table 1.

On each collection, we first use LQE to generate a list of K expanded queries ($K = 30$) for each original query. We append to this list the original query - in the case when all generated reformulations are bad, the reranking approach has a chance to choose not to reformulate. We then use our approach to rerank this list and compare its performance with that of the intial list as well as original query.

3.2 Training Data

We run LQE with the MSN log to obtain a list of reformulations for each original query. We use all these queries to do retrieval and record their *MAP* and use them to create our dataset. Training and testing are done using 5-fold cross validation on this dataset. λ_1 and λ_2 are learned using AdaRank [7] to maximize the average *NDCG@2*. The algorithm ends up choosing either $(\lambda_1 = 1, \lambda_2 = 0)$ or $(\lambda_1 = 0, \lambda_2 = 1)$ depending on the collection.

3.3 Reranking Effectiveness

We use *NDCG@2* to measure the quality of the ranked list of reformulations given by our approach. Reformulations are graded on a scale from *zero* to *four* with respect to the improvement m they provide over the original query. In particular, improvement larger than 0.03 corresponds to a 4, or $(m > 0.03) \rightarrow 4$. Similarly, $(0.01 < m \leq 0.03) \rightarrow 3$, $(0 < m \leq 0.01) \rightarrow 2$, $(m = 0) \rightarrow 1$ and $(m < 0) \rightarrow 0$.

Table 2 summarizes the result: the list of reformulations ranked by our approach has a much higher average *NDCG@2* than the initial list. All improvements are statistically significant at $p < 0.05$ using a two-tailed t-test.

3.4 Retrieval Effectiveness

We define the *MAP* of a ranked list of reformulations as the best *MAP* observed among its top *two* queries. In this section, we compare the *MAP* obtained by (i) the original query, (ii) the list of reformulations generated by LQE, and (iii) the list reranked by our method.

As can be seen in Table 3, the best of the top two reformulated queries ranked by our approach is almost always significantly better than the original query. This is not the case in LQE. In many cases, our method also provides significant improvements over LQE. This result suggests that the reranking can push better reformulations to the first two positions in the ranked list.

4. CONCLUSIONS

In this paper, we have shown that by reranking the list of reformulations generated by the log-based query expansion technique [2] with only two features, we can push more good reformulations into the first two positions in the list. This is reflected in the huge gain of *NDCG@2* and statistically significant improvement in retrieval effectiveness. In the future, we will investigate more features. We hope this will lead to greater improvement in *NDCG@1*, helping retrieval systems to reformulate queries implicitly without user involvement.

5. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #IIS-0711348, and in part by ARRA NSF IIS-9014442. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

6. REFERENCES

[1] S. Cronen-Townsend, Y. Zhou, and W.B. Croft. Predicting Query Performance. In *Proc. of SIGIR*, pages 299-306, 2002.

[2] V. Dang and W.B. Croft. Query Reformulation Using Anchor Text. In *Proc. of WSDM*, pages 41-50, 2010.

[3] R. Jones, B. Rey and O. Madani. Generating Query Substitutions. In *Proc. of WWW*, pages 387-396, 2006.

[4] G. Kumaran and V.R. Carvalho. Reducing Long Queries Using Query Quality Predictors. In *Proc. of SIGIR*, pages 564-571, 2009.

[5] *Proc. of the 2009 workshop on Web Search Click Data*, Barcelona, Spain. ACM New York, NY, USA, 2009.

[6] X. Wang and C. Zhai. Mining Term Association Patterns from Search Logs for Effective Query Reformulation. In *Proc. of CIKM*, pages 479-488, 2008.

[7] J. Xu and H. Li. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proc. of SIGIR*, pages 391-398, 2007.

[8] Y. Zhao, F. Scholer, and Y. Tsegay. Effective Pre-retrieval Query Performance Prediction Using Similarity and Variability Evidence. In *Proc. of ECIR*, pages 52-64, 2008.