An Exploration of Query Term Deletion

Hao Wu and Hui Fang

University of Delaware, Newark DE 19716, USA
haowu@ece.udel.edu, hfang@ece.udel.edu

Abstract. Many search users fail to formulate queries that can well represent information needs. As a result, they often need to reformulate the queries in order to find satisfying results. Query term deletion is one of the commonly used strategies for query reformulation. In this paper, we study the problem in the context of TREC session track. We first show that the reformulated queries used in the session track would lead to less satisfying search results than the original queries, which suggets that the current simulation strategy used in the session track for term deletion might not be a good method. Moreover, we also examine the potential of query term deletion, propose two simple strategies that automatically select terms for deletion, and discuss why the simulation strategy for term deletion fails to reformulate better queries.

1 Introduction

It is often difficult for search users to describe their information needs using keyword queries. Given an information need of a user, the first query formulated by the user may not be effective to retrieve many relevant documents, and the user often needs to reformulate queries in order to find more satisfying search results. Existing studies on query log analysis show that around half of the queries are re-formulated by users [2, 1].

Query deletion is one of the commonly used query reformulation strategies. Intuitively, query deletion can benefit over-specified queries so that the returned search results would have greater coverage of relevant information. Moreover, query deletion can also benefit ill-specified queries so that non-relevant documents that are attracted by noisy query terms would not be returned. A common belief is that search users are capable of selecting terms for deletion and the reformulated queries would lead to more satisfying search results. Thus, existing work on query deletion mainly focused on how to predict which terms would be deleted by search users during query reformulation [1].

In this paper, we focus on the problem of query term deletion in the context of TREC query session track. Specifically, we first analyze the query sessions in the TREC 2010 query session track, and compare the retrieval performance of the original queries with those of the reformulated queries through query term deletion. We find that query reformulation based on term deletion hurt the performance. The failure analysis suggests that the simulation used by the session track tends to delete more terms than necessary.

We also conduct a set of experiment to study the potential of query term deletion. Specifically, we aim to answer the following question: if the decision on which terms to be deleted could be made correctly, would it be possible to improve the retrieval performance with the reformulated queries? Experiment results suggest that query term deletion, if done appropriately, can significantly improve the retrieval performance.

Motivated by the potential of query term deletion on improving retrieval performance, we explore two simple strategies that aim to automatically delete query terms. The first strategy aims to delete less important terms while the second one aims to delete query terms that over-specify the information needs. Empirical evaluation shows that both strategies are effective in improving retrieval performance.

2 Analysis of Term Deletion in Query Sessions

In this section, we conduct experiments to study whether search users are capable of selecting appropriate query terms for deletion. We use the collection provided in TREC 2010 session track [2]. The data set we used is ClueWeb 09 Category B collection. There are 150 topics and each of them have two queries: one is original query and the other is reformulated query. Out of these queries, 26 topics are reformulated through query term deletion and 22 have relevant judgments. Our experiments are based on these 22 topics. The retrieval function is Dirichlet Prior method [3] with $\mu = 500$. The retrieval performance is evaluated based on $nDCG$@20.

Table 1. An example for ideal query term deleting

Candidate queries after term deletion	Retrieval Performance ($nDCG$@20)
disneyland	0.026
special	0.000
offers	0.000
disneyland special	0.017
disneyland offers	0.166
special offers	0.000

Table 2. Comparison among different versions of queries

Queries	Performance($nDCG$@20)	Average Query Length
Original	0.160	4.455
Reformulated	0.138	2.455
Ideal	0.252	3.091

We conduct two sets of experiments. First, we compare the retrieval performance for the original and reformulated queries in order to see whether simulated search users are good at term deletion. Second, we examine the potential of query term deletion. The goal is to study, if done appropriately, whether query term deletion can lead to better retrieval performance. Specifically, for every query, we generate all possible candidate queries after term deletion, evaluate the retrieval performance for each of the queries, and choose the one with best performance.

For example, consider query "disneyland special offers", the value of $nDCG@20$ is 0.103. Table 1 lists all the possible candidate queries after term deletion and their corresponding performance. The candidate query with best performance is referred to as the ideal query in the paper.

Table 2 shows the performance comparison between using the original queries, reformulated queries and ideal queries. We can see that simulated search users on average would delete two query terms, which is larger than that for ideal queries. Moreover, we make two interesting observations.

- The retrieval performance for the reformulated queries is worse than that for the original queries. This observation suggests that simulated users may not capable of deleting query terms that hinder retrieval performance. Thus, the simulation strategy used in the session track might not be a good one.
- If we could have an oracle that always make the correct decisions on which terms to delete, the retrieval performance can be improved significantly.

Clearly, it would be interesting to study what are effective strategies for query term deletion.

3 Methods for Automatic Term Deletion

In this section, we study the methods for automatic query term deletion.

When formulating a query, users may include terms that are less effective in identifying relevant information. Thus, one possible solution is to delete query terms based on their importance, which is referred to as **Imp**. One common measure of term importance is the IDF value of a term, so we can delete query terms with lower IDF values.

Over-specifying an information need is a common reason for query term deletion. Thus, we need to delete terms that causes over-specification without changing the information need. One possible solution is to delete terms that are more semantically related to other terms, which is referred to as **Sim**. The rational is that deleting this kind of terms would hurt the performance less because the related terms are still able to bring a lot of relevant information. Following previous studies [4], we compute the term semantic similarities based on the expected mutual information of a term pair over the document collection. Thus, in the second method, we delete query terms with higher average mutual information values.

We have discussed two strategies for query term deletion. One is to keep deleting query terms with lowest IDF, and the other is to keep deleting terms with highest average MI values. The remaining challenge is to decide when to stop term deletion. We explore two strategies. In the first strategy, we use a threshold as a cut-off. It means that we delete terms whose IDF values are lower than a threshold or delete terms whose MI values are higher than a threshold. In the second strategy, we use the percentage of the number of original query terms as a cut-off. It means that we stop term deletion when the percentage of the query terms that are deleted is larger than the cut-off.

Since we have two term deleting strategies and two stop criteria, there are four methods to be compared. Table 3 shows the results of performance comparison.

Table 3. Performance Comparison

Imp + threshold	Cut-offs	$IDF < 0$(no deletion)	$IDF < 0.5$	$IDF < 1.0$	$IDF < 1.5$	$IDF < 2.0$
	$nDCG$@20	0.160	**0.168**	**0.167**	0.154	0.148
Imp + percentage	Cut-offs	0% (no deletion)	20% terms	30% terms	40% terms	50% terms
	$nDCG$@20	0.160	**0.177**	**0.176**	0.145	0.134
Sim + threshold	Cut-offs	$MI > \infty$ (no deletion)	$MI > 2.5$	$MI > 2.0$	$MI > 1.5$	$MI > 1.0$
	$nDCG$@20	0.160	0.160	**0.163**	**0.179**	0.098
Sim + percentage	Cut-offs	0% (no deletion)	20% terms	30% terms	40% terms	50% terms
	$nDCG$@20	0.160	**0.163**	0.156	0.150	0.106

We see that both term deletion strategies can improve retrieval performance if we choose proper parameters. When deleting terms based on term importance (i.e., IDF), using the percentage of query terms as a cut-off is more effective. The reason is that longer queries tend to include more less important terms. Thus, if we delete terms based on their importance, the number of deleted terms should be proportional to the original query length, i.e., we need to delete more terms for longer queries and fewer terms for shorter queries. When deleting terms based on the semantic similarities (i.e., MI), using the threshold yield to better performance, because not all queries are over-specified and it is unfair to delete the same percentage of terms for every query. In summary, deleting 20% of the original query terms seems to always improve the retrieval performance. This is consistent with the results in Table 2. In particular, we suggest to delete 20% of the original query terms with lowest IDF values because it gives good performance and the parameter value is not collection-dependent.

4 Why did simulated users fail?

In this section, we discuss why the simulation used in the session track fails to choose appropriate terms for deletion.

Table 4. Comparison between reformulated queries and the ideal queries

	terms deleted by the simulation	terms deleted by "oracle"
Total Number	44	30
Average IDF	2.484	2.373
Average MI	1.150	1.335

We first look into what kind of terms are deleted by the simulation during reformulation and what kind of terms should be deleted. Table 4 shows the term statistics in these two situations. We see that simulated users also tend to delete terms with lower IDF and high MI values. But simulated users often delete more terms than necessary.

We then conduct per-query analysis to compare query formulated by simulation with the ideal queries, and classify simulated user mistakes into three categories: (1) *over-deletion*: delete more terms than necessary; (2) *under-deletion*: delete fewer terms than necessary; (3) *error-deletion*: delete wrong terms. Table 5 shows our summary about these three error types. It is interesting to see that the number of over-deletion errors is high. Since it is much easier to make error-deletion than over-deletion or under-deletion, it shows that the simulated users

Table 5. Analysis of Errors

Error types	Num. of queries	Examples		
		Original queries	**Ideal queries**	**Users' reformulated queries**
Over-deletion	9	diabetes education videos books	diabetes education videos books	diabetes education
Under-deletion	1	windows defender reports software problems	defender problems	windows defender problems
Error-deletion	8	wall hung toilet kohler	hung toilet kohler	wall hung toilet

has ability to differentiate the usefulness of query terms, but they delete terms too aggressively.

To summarize our finding so far, the simulation is good at deciding which terms to delete but they tend to delete more terms than necessary. This is probably the main reason why the reformulated queries perform worse then the original ones. To further verify our hypothesis, we conduct another set of experiments. We force the system to delete the same number of terms as the simulation's deletion for every query. The results are shown in Table 6. It is clear that the human are smarter at selecting which terms to delete than the system based on the two proposed term deletion methods. Thus, we expect that if simulated search users are more conservative in deleting query terms, the reformulated queries should lead to more satisfying search results.

Table 6. Performance comparison when deleting the same number of terms as the simulation did for every query

	simulation's deletion	deleting terms with lowest IDF	deleting terms with highest MI
$nDCG@20$	0.138	0.107	0.120

5 Conclusions and Future Work

We study the problem of query deletion in the context of TREC query session track. We propose two simple strategies for automatic term deletion, and experiment results show that both methods are effective if the parameter values are set appropriately. We also find that the poor query reformulation of simulated sessions is caused by the fact that the simulation tends to delete more terms than necessary while it is good at picking terms that need to be deleted.

In the future, we plan to conduct similar studies on real query sessions and over multiple collections. We also plan to explore other strategies for automatic query term deletion.

References

1. R. Jones and D. C. Fain. Query word deletion prediction. In *Proceedings of SIGIR'03*, 2003.
2. E. Kanoulas, B. Carterette, P. Clough, and M. Sanderson. Session track overview. In *Proceedings of TREC'10*, 2010.
3. C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR'01*, 2001.
4. W. Zheng and H. Fang. Query aspect based term weighting regularization in information retrieval. In *Proceedings of ECIR'10*, 2010.