

Automatic Query Reformulation with Syntactic Operators to Alleviate Search Difficulty

Huizhong Duan
University of Illinois at Urbana-
Champaign
201 N Goodwin Ave
Urbana, IL 61801 USA
duan9@illinois.edu

Rui Li
University of Illinois at Urbana-
Champaign
201 N Goodwin Ave
Urbana, IL 61801 USA
ruili1@illinois.edu

ChengXiang Zhai
University of Illinois at Urbana-
Champaign
201 N Goodwin Ave
Urbana, IL 61801 USA
czhai@cs.uiuc.edu

ABSTRACT

Modern search engines usually provide a query language with a set of advanced syntactic operators (e.g., plus sign to require a term's appearance, or quotation marks to require a phrase's appearance) which if used appropriately, can significantly improve the effectiveness of a plain keyword query. However, they are rarely used by ordinary users due to the intrinsic difficulties and users' lack of corpora statistics. In this paper, we propose to automatically reformulate queries that do not work well by selectively adding syntactic operators. Particularly, we propose to perform syntactic operator-based query reformulation when a retrieval system detects users encounter difficulty in search as indicated by users' behaviors such as scanning over top k documents without click-through. We frame the problem of automatic reformulation with syntactic operators as a supervised learning problem, and propose a set of effective features to represent queries with syntactic operators. Experiment results verify the effectiveness of the proposed method and its applicability as a query suggestion mechanism for search engines. As a negative feedback strategy, syntactic operator-based query reformulation also shows promising results in improving search results for difficult queries as compared with existing methods.

Categories and Subject Descriptors

H.3.3 [INFORMATION STORAGE AND RETRIEVAL]:
Information Search and Retrieval – *query formulation*

General Terms

Algorithms, Performance, Experimentation

Keywords

Query reformulation, syntactic operator, search difficulty.

1. INTRODUCTION

Query languages of modern search engines usually include a set of advanced syntactic operators to supplement traditional keyword query [1, 2]. For instance, a necessity operator (plus sign) preceding a query term requires the term to be present in each

relevant document; a phrase operator (a pair of quotation marks) imposes that relevant documents must contain the phrase consisting of the quoted terms. To distinguish from keyword query, we refer to a query with syntactic operators as a *syntax query*. For example, Figure 1a shows a keyword query, while Figure 1b and Figure 1c show two *syntax queries* which have the same set of terms as the keyword query shown in Figure 1a. For convenience of discussion, we further denote this type of syntactic operator-based reformulation as *syntactic reformulation*.

If used appropriately, syntactic reformulation can be very effective for improving retrieval accuracy, turning an otherwise ineffective query to an effective one. Figure 1 shows an example of using syntactic reformulation to improve the top ranked results with a major US search engine¹. As we can see from Figure 1a, none of the top ranked documents is relevant to the query. In contrast, in Figure 1b, by using the syntactic query with the necessity constraint on term “unix”, we are able to find two (2nd and 3rd) relevant documents out of the top three. This is because the search engine overlooked the term “unix” in the original query, which is typically due to the coarse estimation of term importance. By imposing the necessity constraint, the query emphasizes the importance of the term and re-ranks the results according to whether it is contained in each document. In Figure 1c, with an even more complicated syntactic reformulation, we further impose a phrase constraint on “default java”. This effectively eliminates the possible ambiguities of the query caused by matching terms separately. As a result, all the top three ranked documents are relevant documents. The proper use of syntactic operators not only clarifies user's information need, but also gives clues to the retrieval system on how to optimize the search results.

However, very few users make use of the syntactic operators in their daily search activities. Statistics² from a search engine query log show that only less than 0.5% queries used syntactic operators. This is either because users are unfamiliar with the semantics of the operator, or because they lack the appropriate knowledge and statistics to formulate working syntax queries.

In this paper, we propose to help users take advantage of the rich query syntax by automatically formulating potentially effective syntax queries based on keyword queries. Particularly, we propose to automatically perform the query reformulation with syntactic operators when users encounter difficulty in search. Such

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.
Copyright 2011 ACM 978-1-4503-0717-8/11/10...\$10.00.

¹ <http://www.google.com>

² Based on a sample of MSN query log in 2006.

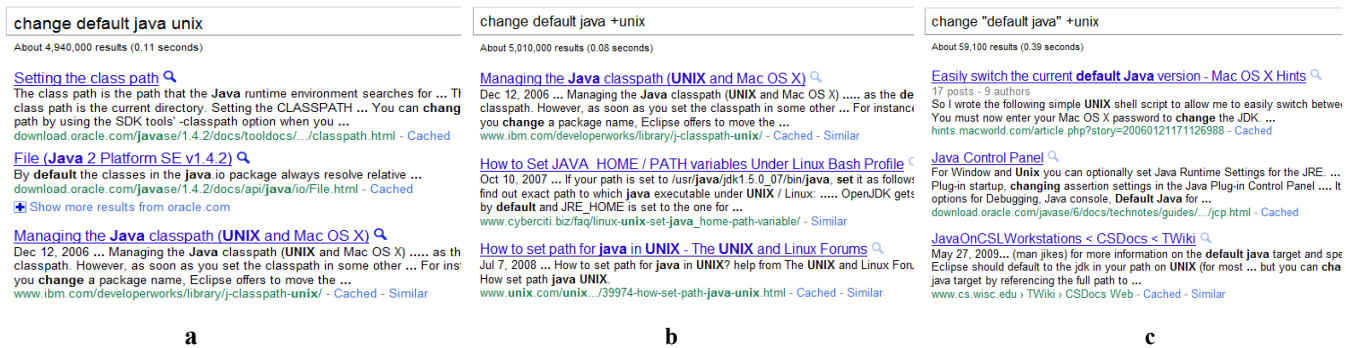


Figure 1. Example of Using Syntactic Operators for Improving Search Result.

difficulty could be detected through monitoring users' behaviors. For instance, one possible indicator could be when a user skipped a page of top ranked results to the next page. Our assumption is that if a user can find relevant documents in the top ranked results, she is likely satisfied with the original ranking and thus there is no much need for syntactic reformulation. We thus focus on studying how to exploit query syntax reformulation to help users when it is evident that the user cannot find any relevant document in the top k results. This scenario is similar to the case of negative relevance feedback [11, 12] in the sense that there is a common goal of re-ranking the unseen results based on a small number of negative examples already encountered by the user. However, while existing negative feedback techniques primarily rely on word frequency analysis in the negative examples to improve the query representation [11], we study how to leverage query syntax operators to improve the syntax of a query, which is a different and complementary strategy and has two advantages over the existing methods for negative feedback: first, it has better interpretability to users. i.e., the suggested reformulated query can be more easily verified by a user if deployed in an interactive feedback system; second, while most negative feedback methods "blindly" treat a whole document as non-relevant, our method would attempt to "diagnose" the cause of poor results and correct any missing semantic constraint via appropriate syntactic operator.

To perform automatic syntactic reformulation, we adopt a supervised learning framework. Particularly, we cast the problem as to predict the potential benefit of each candidate syntax query in improving the performance of a given keyword query. If we are confident on the potential improvement, we can suggest the reformulation to the user or use it directly to refine the search result. One problem in training the prediction model is that syntactic reformulations of different keyword queries may not have comparable performances. To circumvent this problem, we make use of the learning to rank techniques.

We propose and study three types of features to represent the characteristics of a syntax query. Query difficulty measures the intrinsic difficulty of syntax query. Distinguishability computes the amount of changes a syntax query brings to the original result. Negativity measures the similarity between the candidate query and the negative relevance feedback. The three types of features are all generalized features that can be computed for any filtering-based operators. Experiments demonstrate their effectiveness in formulating highly beneficial syntax queries.

2. RELATED WORK

Query reformulation/refinement is a broad topic on using modified versions of queries to improve search results. Our work is naturally subsumed by this topic. Under this broad topic, query

expansion, query contraction, spelling correction have been extensively studied. Guo et al. [5] proposed a modified CRF model for simultaneously performing spelling correction and query segmentation. Yet it is unclear whether the proposed method is applicable to different types of operators, especially in alleviating search difficulty. In this paper, we conduct systematic study on query reformulation with syntactic operators to alleviate search difficulty. Besides explicit reformulation of query, there are also studies on implicit refinement of queries, such as term proximity [8, 9] and term necessity [13] refinement. This type of refinement is usually transparent to users and makes more subtle changes to original results. Compared with these work, the explicit use of syntactic operator has the advantage of being more interpretable and easy to implement, especially across different retrieval systems.

Our research is also strongly related to the work on negative feedback. Wang et al. used EM algorithm to estimate language models for negative feedback and use them to re-rank the rest of the results [11, 12]. Our method also uses the negative feedback information to assist search difficulty, but it distinguishes itself by making use of syntactic operators to introduce additional semantics to the query. Compared with existing methods, this is a novel complementary approach and potentially provides better usage of the negative feedback.

3. AUTOMATIC REFORMULATION WITH SYNTACTIC OPERATORS

We formulate the automatic syntactic reformulation of keyword queries as a supervised learning problem. Particularly, we cast the problem as to predict the potential performance improvement from each candidate syntax query over a given keyword query.

Formally, given a keyword query q , a syntactic operator op and a target performance metric M , our goal is to find a list of syntactic reformulations of q through the use of op , denoted as $S_{op}(q) = \{q_1, q_2, \dots, q_n\}$, which is ranked according to M in descending order: $M(q_1) > M(q_2) > \dots > M(q_n)$. When it is required to output top m suggestions, the system will respond with a list consisting of q_1, q_2, \dots, q_m . When a syntactic reformulation is required to directly refine the search results, the system will output the top ranked query q_1 if $M'(q_1) = M(q_1) - M(q) > 0$, or otherwise the original query q . The training process is to learn a function f , which takes a set of features of a syntax query q_i and predict the performance improvement $M'(q_i)$.

Since there are an exponential number of possible syntax queries for each syntactic operator, we limit the system to consider single appearance of each operator. Although we only explore the limited scope of use of these operators, the proposed methodology

is general and can be potentially applicable to other filtering-based operators and multiple uses of each operator.

To solve the learning problem, a natural way is to use a regression model. In this method, a model will be learned to predict the performance for any possible syntax query. One problem with this method is that the performances of syntax queries reformulated based on different keyword queries are usually not directly comparable. To circumvent the problem, we adopt the learning to rank framework. In this setting, syntax queries generated from each keyword query are considered as a group; the loss function is defined on the ranking order of members in each group instead of on the absolute value of the performance. This loss function is more natural in our problem setting and avoids the issue of incomparability. We then use the learned model to predict the potential benefit in performance for each candidate in syntactic reformulation. Particularly, we adopt Ranking SVM [7] as our learning method.

3.1 Features

We propose three types of features, namely *difficulty*, *distinguishability* and *negativity*. All three types of features are defined in a general way so that they can be computed for any type of filtering-based operators. As used in previous discussions, we denote q as the keyword query, op as the target operator and q_x as a syntactic reformulation of q with operator op .

3.1.1 Difficulty

The difficulty feature aims to measure the intrinsic difficulty of a syntax query. Query difficulty prediction has been widely studied in recent years [4, 6]. We modify the clarity feature [4] slightly to make it applicable to syntax queries:

$$\begin{aligned} Clarity(q_x) &= KL(\theta_m || \theta_c) \\ &= \sum_{w \in V} p(w|\theta_m) \log \frac{p(w|\theta_m)}{p(w|\theta_c)} \end{aligned} \quad (1)$$

where θ_m is the language model estimated from the set of matched documents S_m of q_x , θ_c is the language model estimated from the entire collection. Queries with high clarity are more likely to work well.

In addition to clarity, we propose another difficulty feature, which is inspired by the inverse document frequency (IDF). We generalize this concept to compute the specificity of a syntax query:

$$GIDF(q_x) = \log \left(\frac{N_c - N_{S_m} + 0.5}{N_{S_m} + 0.5} \right) \quad (2)$$

where N_c and N_{S_m} are the number of documents in the entire collection and matched set, respectively. Intuitively, a query matching fewer documents is more specific and meaningful.

3.1.2 Distinguishability

The idea of the distinguishability feature is to quantify the changes a syntax query brings to the original query. For this purpose, we define cross clarity between θ_m and θ_q , the language model estimated from the search results of q :

$$CrossClarity(q_x, q) = KL(\theta_m || \theta_q) \quad (3)$$

This feature measures the change in the topical formation of the syntax query q_x and the original query q .

Besides measuring content changes, we use another feature to measure the change in the ranking of documents. Particularly, we

measure the correlation between the document rankings of q and q_x :

$$Cor(q_x, q) = \frac{\#concord(q_x, q) - \#discord(q_x, q)}{\frac{1}{2}N_q(N_q - 1)} \quad (4)$$

where $concord(q_x, q)$ and $discord(q_x, q)$ are the sets of concordant pairs and discordant pairs between the two ranking lists of search results of q_x and q . The correlation feature quantifies the changes q_x brings to the original ranking of q .

3.1.3 Negativity

The negativity feature measures the similarity between the syntax query q_x and the negative documents. A query less similar to the negative documents naturally has higher chances of working well.

Again, we define the content based negativity feature as the cross clarity between the language model estimated from the matched documents S_m and the language model estimated from the negative feedback documents S_n .

In addition, we define another negative feature as the generalized inverse negative frequency (GINF):

$$GINF(S_m, S_n) = \log \left(\frac{N_{S_n} + 0.5}{N_{S_m \cap S_n} + 0.5} \right) \quad (5)$$

In effect, this feature indicates the necessity of requiring a particular operator in the original keyword query.

By computing negativity features with different operators, we are actually “diagnosing” why the original query does not work well.

3.2 Combining Operators in Prediction

Our proposed algorithm not only works for predicting each operator separately, but can also be applied to predict different operators jointly, as filters can be applied additively. We refer to this joint prediction method as *Operator-Combination*.

An alternative method is *Result-Combination*, in which we predict each operator separately and select the reformulation with the best predicted performance. In practice we find this method not only more effective but also more efficient than the Operator-Combination, as it considers much fewer candidates in prediction.

4. EXPERIMENTS

4.1 Experiment Setup

We use TREC 2004 Robust track [10] as our experiment dataset. The title field and description field of the queries are used to represent (relatively) short query and (relatively) long query, respectively. Not all the queries are difficult queries. To simulate the scenario of search difficulty, we adopt the minimum deletion method proposed in [11]. We use BM25 [3] as our baseline. In addition, we also implement *MultiNeg*, a state-of-the-art method for using negative feedback [11]. It is worth noting that our method does not conflict with existing methods for using negative feedback. Instead, we use negative feedback in a different manner, which could potentially bring complementary factor to the existing methods. To test this idea, we further combine the use of syntactic reformulation with the MultiNeg method.

We evaluate automatic syntactic reformulation with necessity and phrase operator by using them to directly refine search result by re-ranking unseen documents. NDCG@10 is used as the primary metric. All the reported results are based on 5-fold cross validation. Statistical significance is indicated by * (p-value<0.05).

4.2 Experiment Results

Table 1 and Table 2 show the performances on long queries (description) and short queries (title) respectively. We see that for long queries, reformulation with necessity operator achieves higher performances than with phrase operator. However, for short queries, phrase operator tends to work better. Long queries are much more verbose and noisy, and thus more likely have the central topic missed in returned documents. Reformulation with necessity operator is more useful here as it discovers the important but underrepresented topical term and ensures it to be matched. A short query has fewer keywords and therefore less noise. However, the connection between keywords is usually lost as a cost of being succinct. Reformulation with phrase operator alleviates the problem by imposing the phrase constraint strongly connected terms. We also see the Result-Combination strategy always outperforms Operator-Combination. Result-Combination usually brings further improvement to the performance. More importantly, it provides a more robust solution compared with reformulation with single type of operator.

Table 1. Automatic Syntactic Reformulation to Directly Refine Search Results for Long Queries (Description)

Run Name	NDCG@10	P@1	MAP
Baseline	0.065	0.157	0.089
Necessity	0.077*	0.205*	0.104*
Phrase	0.069	0.161	0.093
Operator-Combination	0.065	0.161	0.088
Result-Combination	0.079*	0.209*	0.114*
MultiNeg	0.074*	0.216*	0.093
RC+MultiNeg	0.081*	0.221*	0.115*

Table 2. Automatic Syntactic Reformulation to Directly Refine Search Results for Short Queries (Title)

Run Name	NDCG@10	P@1	MAP
Baseline	0.078	0.217	0.111
Necessity	0.081	0.229	0.115
Phrase	0.083	0.221	0.119*
Operator-Combination	0.076	0.201	0.108
Result-Combination	0.082	0.225	0.119*
MultiNeg	0.078	0.216	0.111
RC+MultiNeg	0.084*	0.225	0.119*

Result-Combination outperforms MultiNeg. MultiNeg method takes the entire content of a negative document as non-relevant. In our method, we try to find the commonly missing semantics among negative examples. Compared with MultiNeg, this is a more precise way of using negative feedback and is therefore more effective. The performance could be further improved by combining syntactic reformulation with MultiNeg (RC-MultiNeg). Since our method does not directly use the negative information to refine scores of documents, it is complementary with the existing methods that work in this way.

We see that it is generally more difficult to improve the search results for short queries. Short queries are typically more succinct. As a result, there is less room for applying syntactic operators.

5. CONCLUSION AND FUTURE WORK

The ability to help users reformulate effective queries when their original queries don't work well is a crucial criterion for a good retrieval system. In this paper we propose and study a novel way of automatic query reformulation through the use of query syntax operators. With appropriate use of syntactic operators, we are able to detect the "missing semantics" in the original query and amend it in the reformulated query.

We formulate automatic syntactic reformulation as a supervised learning problem under the framework of learning to rank. We propose a set of effective features to represent the characteristics of syntax queries. The proposed approach is general; it can be applied to syntactic reformulation with any filtering-based operators, and can be directly incorporated into an existing search engine to improve user's experience with difficult queries.

Extensive experiments are conducted to test the effectiveness of the proposed methods. For the task of directly refining future search results with the reformulated query, we observe promising results in improving the performance. It is demonstrated that with proper strategy, we could combine the reformulations with different operators to achieve a robust solution. Our method is also complementary with existing methods for using negative relevance feedback as further performance improvements are observed when they are combined in use.

In future, we plan to explore the use of more useful syntactic operators for query reformulation to alleviate search difficulty.

6. REFERENCES

- [1] <http://www.google.com/support/websearch/bin/answer.py?hl=en&answer=136861>
- [2] <http://msdn.microsoft.com/en-us/library/ff795620.aspx>
- [3] http://en.wikipedia.org/wiki/Okapi_BM25
- [4] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. Predicting query performance. In *SIGIR '02*, 2002.
- [5] Jiafeng Guo, Gu Xu, Hang Li, and Xueqi Cheng. A unified and discriminative model for query refinement. In *SIGIR '08*, 2008.
- [6] Claudia Hauff, Djoerd Hiemstra, and Franciska de Jong. A survey of pre-retrieval query performance predictors. In *CIKM '08*, 2008.
- [7] Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02*, 2002.
- [8] Yves Rasolofo and Jacques Savoy. 2003. Term proximity scoring for keyword-based retrieval systems. In *ECIR'03*, 2003.
- [9] Tao Tao and ChengXiang Zhai. An exploration of proximity measures in information retrieval. In *SIGIR '07*, 2007.
- [10] Ellen M. Voorhees. Overview of the trec 2004 robust retrieval track. In *TREC '04*, 2004.
- [11] Xuanhui Wang, Hui Fang, and ChengXiang Zhai. A study of methods for negative relevance feedback. In *SIGIR '08*, 2008.
- [12] Xuanhui Wang, Hui Fang, and ChengXiang Zhai. Improve retrieval accuracy for difficult queries using negative feedback. In *CIKM '07*, 2007.
- [13] Le Zhao and Jamie Callan. Term necessity prediction. In *CIKM '10*, 2010.