Query Reformulation Using Anchor Text

Van Dang
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
vdang@cs.umass.edu

W. Bruce Croft
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
croft@cs.umass.edu

ABSTRACT

Query reformulation techniques based on query logs have been studied as a method of capturing user intent and improving retrieval effectiveness. The evaluation of these techniques has primarily, however, focused on proprietary query logs and selected samples of queries. In this paper, we suggest that anchor text, which is readily available, can be an effective substitute for a query log and study the effectiveness of a range of query reformulation techniques (including log-based stemming, substitution, and expansion) using standard TREC collections. Our results show that log-based query reformulation techniques are indeed effective with standard collections, but expansion is a much safer form of query modification than word substitution. We also show that using anchor text as a simulated query log is as least as effective as a real log for these techniques.

Categories and Subject Descriptors

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

General Terms

Algorithms, Measurement, Performance, Experimentation.

Keywords

Query substitution, query expansion, query reformulation, query log, anchor text, anchor log.

1. INTRODUCTION

When users interact with a search engine, they not only find documents satisfying their information need, they also provide the search engine with implicit feedback about the results returned by the system. The search engine keeps track of this information in the form of a query log, which basically includes queries submitted by users and documents from the result pages that have been clicked on to view.

Since a query log is a rich information source about users, the analysis of these logs has recently become an active research area. Query logs have been used for many tasks, one of which is query reformulation [23, 13, 3, 25, 27]. Query reformulation aims to solve the vocabulary mismatch problem in Information Retrieval (IR) by changing the original query to a form that is a better match with relevant documents. Much of the previous work operates on the whole query level, which is more commonly known as query recommendation. This class of techniques first clusters similar queries based on commonly clicked documents [3, 25] or the similarity of vocabulary used in clicked documents [27], and uses queries in the same cluster as recommendations for one another.

The techniques that we focus on generate new queries by substituting or adding words or phrases to the original query. For example, the technique proposed by Jones et al. [13] works on the phrase level by looking at successive pairs of queries in user sessions. Two queries that are different from each other by only one phrase are selected and the corresponding pairs of phrases are recorded as substitution candidates, which are used to generate substitutions for new queries. This study, however, did not evaluate the retrieval effectiveness of the new queries with substitutions. Another example of recent work on log-based query reformulation is that of Wang and Zhai [23], which works on the word level. The method first extracts term associations based on their context distribution. For a new query, the method will decide whether to substitute a term with one of its "similar" words based on whether this new word matches the context of the query better than the original term. This method has been shown to be effective with web queries using click data as relevance judgments.

Context sensitive stemming based on query logs is another type of query reformulation that has been studied [20]. Instead of reducing words into their root forms at indexing time, the stemmer determines at query time which word variants should be used to expand the query. Although it was demonstrated in the work of Peng et al. [20] that context sensitive stemming is useful for web search, they evaluate it on a sample of web queries and do not compare it against traditional stemming approaches.

These and other studies of query reformulation based on query logs nearly all make use of nonstandard approaches to evaluation and proprietary query logs. This makes the comparison of techniques difficult and their applicability to new applications and domains unclear. In this paper, we compare reformulation techniques using TREC collections and suggest modifications based on the results. In particular, we find that expansion is a more reliable form of modification than substitution.

To address the problem of the lack of availability of query logs, we instead propose to use anchor text to simulate the important parts of a log. Anchor text is well-known to be an important feature for effective web search. Previous researchers have also noted the similarity of anchor text to queries. For example, Nallapati et al. [19] used anchor text as relevant queries to train a retrieval model. Kraft and Zien [14] demonstrated that using anchor text to refine a query is better than doing so with the document collection. There have also been studies showing that anchor text resembles real queries in terms of term distribution and length [11]. Given these results, in this paper we construct a simulated query log or *anchor log* from the anchor text in a web test collection. We then evaluate the log-based query reformulation techniques using both the anchor log and a real query log (the MSN query log [18]) and show that the anchor log produces results that are at least as effective.

In the next section, we describe how we construct the anchor log and compare it to the MSN log. Section 3 presents the query substitution technique that we use in more detail. Section 4 describes the context-sensitive query stemming technique. Section 5 contains the experimental results based on the anchor log and query log, as well as a discussion of those results. Section 6 briefly mentions some more related work and Section 7 concludes.

2. BUILDING AN ANCHOR LOG

A query log can contain a broad range of information about user behavior during interaction with a search engine while formulating a query and browsing the search results. The most important parts of the log for query reformulation techniques are the user queries and the URLs (or other identifiers) of documents that are clicked on for those queries. Session information, which records the sequence of queries in a search session, is also used by some techniques.

An anchor log constructed from anchor text is much more limited. Each anchor text is associated with a link to a particular document. Since the anchor text is chosen manually to represent the page, its content is very relevant to the document. Brin and Page [6] point out that the anchor texts often provide more accurate description of the page than the page itself. Thus, in this paper, we simply construct an anchor log that consists of pairs (anchor text, URL) where the anchor text corresponds to a query in a query log, and the URL is the associated link for the anchor text, which corresponds to a click in a query log.

For this paper, we built the anchor log from the TREC Gov-2 web collection (25 million web pages crawled from the .gov domain during early 2004). Only anchors that contain English words and non-stopwords were kept. The same simple filter was applied on the MSN Log. Table 1 compares some basic statistics to the MSN query log, which was a sample of user queries submitted to a search engine over a one month period. Although the size of the anchor log is much larger than the MSN Log, it contains a lot of noise in the form of anchors such as "index", "print version", and "click here". These common anchors could be removed quite easily, but we did not do it for this paper.

One of the limitations of the anchor log is that it does not contain session information. Since the query reformulation technique we evaluate uses session data to filter out unrelated words, we defined a session in the anchor log to be simply the group of anchor texts that point to the same web page. It should also be noted that session information in the MSN log is limited due to the sampling methodology.

Table 1: Statistics of MSN Log and Anchor Log

	MSN Log	Anchor Log
# Total Queries	14,035,893	526,966,029
# Unique Queries	6,087,365	20,838,241
Average Query Length	2.68	2.62

3. CONTEXT SENSITIVE QUERY SUBSTITUTION

This section briefly describes the method proposed by Wang and Zhai [23]. The method first estimates the context distribution for all words in the query log. It then learns a translation model to "translate" from one word to another based on their distributional similarity. Finally, a substitution model is built on the top of the translation model. Given a query term and a query context, the substitution model will decide whether to make a substitution, and if so, which candidates among those suggested by the translation model should be used. Details are as follows.

3.1 Definition of Context

The *i-th left context* (L_i) of a query term w is the set of words that occur at the $i - th$ position away from w on its left hand side in the query. The *i-th right context* (R_i) is defined in a similar manner. For example, if the query is *oscar winner selection*, we say that *oscar* is in the L_1 context of *winner* and *selection* is in the R_1 context.

3.2 Context Distribution

Let C denote a particular type of context (it can be either L_i or R_i) and $C(w)$ denote the set of words that are in the C context of w and $count_w(c_i)$ denote the number of times that the word c_i occurs in the C context of w. Given a term w, the probability distribution of its context words is given by

$$P_C(c_i|w) = \frac{count_w(c_i)}{\sum_{c_j \in C(w)} count_w(c_j)} \tag{1}$$

To avoid the zero-probability problem, a smoothed distribution is also given,

$$\tilde{P_C}(c_i|w) = \frac{count_w(c_i) + \mu P(c_i|\theta)}{\sum_{c_j \in C(w)} count_w(c_j) + \mu} \tag{2}$$

where $P(c_i|\theta)$ is the probability of the word c_i in the entire collection and μ is the Dirichlet prior parameter, which was set to 1500 in all of our experiments.

3.3 Translation Model

The translation model is used to capture words that can be used in the same context. The idea is words that occur in similar context are "related" to one another, and thus can be good substitution candidates for one another.

The translation model uses the KL divergence between w and s to model the probability of "translating" from w to s, denoted as $t(s|w)$. The estimation of $t(s|w)$ is given by

$$t(s|w) = \frac{e^{-D(P_C(.|w)||\tilde{P_C}(.|s))}}{\sum_u e^{-D(P_C(.|w)||\tilde{P_C}(.|u))}} \quad (3)$$

where $D(P_C(.|w)||\tilde{P_C}(.|s))$ is the KL divergence calculated as

$$D(P_C(.|w)||\tilde{P_C}(.|s)) = \sum_{c \in C(w)} P(c|w) log \frac{P(c|w)}{\tilde{P_C}(c|s)} \quad (4)$$

Wang and Zhai [23] used a combination of L_1 and R_1 to estimate the translation probability. In our experiments, however, we observed almost no difference in retrieval performance between using only the L_1 context and the combination. Thus, we used only L_1 in our experiments, which turns out to be identical to the distributional word similarity estimate [21, 10].

As noted by the authors [23], for any given word w the translation model only suggests paradigmatic words. There is no guarantee that they are related in meaning. In an effort to filter out unrelated word pairs, they compute the normalized mutual information between all pairs (s, w) as

$$NMI(s, w) = \frac{MI(s, w)}{MI(w, w)} \quad (5)$$

where $MI(s, w)$ is the mutual information of s and w over the query log sessions. Pairs (s, w) that have NMI below a pre-defined threshold τ are discarded. The idea is unrelated words would not often appear in the same sessions. τ was set to 0.001 in our experiments. However, the elimination of such pairs is not done in this step but in the next.

3.4 Query Substitution Model

Given a query $q = w_1 \dots w_{i-1} w_i w_{i+1} \dots w_n$ where w_i is the candidate for substitution, the substitution model can estimate the probability of substituting the term w_i for any new term s. The substitution probability $P(w_i \rightarrow s|q)$ is given by

$$P(w_i \rightarrow s|q) \propto t(s|w_i) \times P(w_1 \dots w_{i-1}\_w_{i+1} \dots w_n|s) \quad (6)$$

where $P(w_1 \dots w_{i-1}\_w_{i+1} \dots w_n|s)$ can be considered the probability that the new term s fits into the context of the query and is computed as

$$\tilde{P_{L_2}}(w_{i-2}|s) \times \tilde{P_{L_1}}(w_{i-1}|s) \times \tilde{P_{R_1}}(w_{i+1}|s) \times \tilde{P_{R_2}}(w_{i+2}|s) \quad (7)$$

For details about the formula, see [23].

3.5 Candidate Queries Generation

For any given query $q = w_1 \dots w_{i-1} w_i w_{i+1} \dots w_n$, the model will iterate through all terms w_i and try to replace each of them. This results in a set of substituted queries each of which is different to q by only one term.

For each w_i, assuming that the translation model recommends n candidates $s_1, s_2, \dots, s_n$ to replace w_i, the substitution model will determine whether to make the substitution, and if so which candidate to select by following three steps:

1. Consider only the top M translation candidates s_i sorted by their $t(s_i|w_i)$
2. Remove all s_i that have $NMI(s_i, w_i) < \tau$
3. All remaining candidates s_i are considered equally good and $t(s_i|w_i)$ is set to 1.0 for all s_i. Their $P(w_i \rightarrow s_i|q)$ is then computed by formula (6)

The substitution model decides to make the substitutions for each of the remaining s_i that satisfies

$$\frac{P(w_i \rightarrow s_i|q)}{P(w_i \rightarrow w_i|q)} > 1 \quad (8)$$

The idea is that we only make the substitution when the new term is better than the original term. In case the substitution $w_i \rightarrow s_i$ is made, $P(w_i \rightarrow s_i|q)$ is used as the score of this substituted query. In the end of this step, we have a list of N substituted queries sorted by their score.

4. CONTEXT SENSITIVE QUERY STEMMING

We can see that context sensitive query stemming is, in fact, a special form of query reformulation where the candidates are now limited to only variants of query words.

Finding stemming candidates is much easier than finding substitution candidates in query reformulation. It is as simple as finding all variants for each query word. Instead of constructing the translation model, we cluster all unigrams in the Google-ngram corpus into groups based on their root form given by the Porter stemmer. Each word w_i is treated as the stemming variants for all other words w_j within its group. We set $t(s|w)$ to 1 for all pairs of words.

Unlike query substitution, where we generate a set of ranked candidate queries, we generate only one single new query using stemming. The procedure is very simple. We also iterate over all query terms. For each term w_i, we use formula (6) to calculate the score for it and all its variants w_j. We expand the query with all w_j that satisfy

$$\frac{P(w_i \rightarrow w_j|q)}{P(w_i \rightarrow w_i|q)} > \theta \quad (9)$$

We empirically set $\theta = 0.5$ in our experiment to tolerate variability in probability estimation. The entire procedure is, in fact, very similar to that proposed by Peng et al. [20], except that their technique limits the stemming problem to handling pluralization. Since stemming a word in most cases will not change the query intent (at least the risk is much less than that of substituting it with a totally different word), we find that formula (7) is too strict for stemming. Thus, we relax (7) to

$$\prod_{j=i-2, j \neq i}^{i+2} \tilde{P_{LR2}}(w_j|s) \quad (10)$$

where $\tilde{P_{LR2}}(.|s)$ is a distribution over all words co-occurring with s within a distance of two words either to the left or right.

5. EVALUATION

In this section, we compare the effectiveness of the anchor log and the MSN log for the task of query reformulation using substitution and stemming.

5.1 Experimental Setup

Test Collections: All evaluation is done using three standard TREC collections: Robust 04, WT10G and Gov-2. It should be noted that the three collections are quite different; Robust04 is a newswire collection, while the other two are web collections.

Table 2: Summary of three TREC collections

Collection	# Docs	# Query
Robust04	528,155	250
WT10G	1,692,096	100
Gov-2	25,205,179	150

Query Set: We used all queries associated with each collection to evaluate the logs' effectiveness. In particular, we treat all title queries as "short" queries and description queries as "long" queries. Table 2 provides the statistics of the three test collections.
Evaluation Metric: We used precision at 5 (P@5) and precision at 10 (P@10) to measure retrieval effectiveness.

We used Lemur/Indri[1] as the retrieval software and language modeling [8] was used for retrieval. Parameter values were chosen empirically. The Dirichlet prior μ was set to 1500 and the normalized mutual information threshold was $\tau = 0.001$. We set M (the number of translation candidates considered by the substitution model) to 20. Stopword removal was done only at query time in all experiments.

5.2 Query Stemming Experiment

In this section, we compare the retrieval performance given by the context sensitive stemming method using query logs against the baseline where no stemming is done and the conventional stemming approach using the Krovetz stemmer (K-stem).

5.2.1 Experiment Design

We prepared two indexes for each collection: one stemmed using the Krovetz stemmer and one unstemmed. For the baseline approach, the original query is used to do retrieval on the unstemmed collection. The conventional stemming approach first stems the query and then uses it to do retrieval on the stemmed index. The context sensitive stemming approach first learns stemming variants from a given log (either the MSN log and the anchor log) as described in Section 4. It then decides at run-time what stemming variants will be added to the original query to create the query that is used against the unstemmed index.

5.2.2 Results

Fig. 1 shows P@10 for all methods. With short queries, there is no significant difference between different methods. With long queries, however, we can see that stemming using either log works significantly better than the baseline on all three collections. Krovetz stemming, on the other hand, gives significant improvement over the baseline only on WT10G. Significance is measured using the t-test with $p - value \leq 0.05$. We also note that the performance of the anchor log is very similar to that of the MSN log.

5.3 Query Reformulation Experiment

In this section, we will first evaluate the effectiveness of the substitution method on three TREC collections. Then we show that expansion is a more reliable way to do query reformulation, which results in much better overall performance.

[1]http://www.lemurproject.org

Table 3: Statistics of queries used for reformulation

Collection	Short Query	Long Query
Robust04	224	246
WT10G	79	95
Gov-2	136	147

5.3.1 Experiment Design

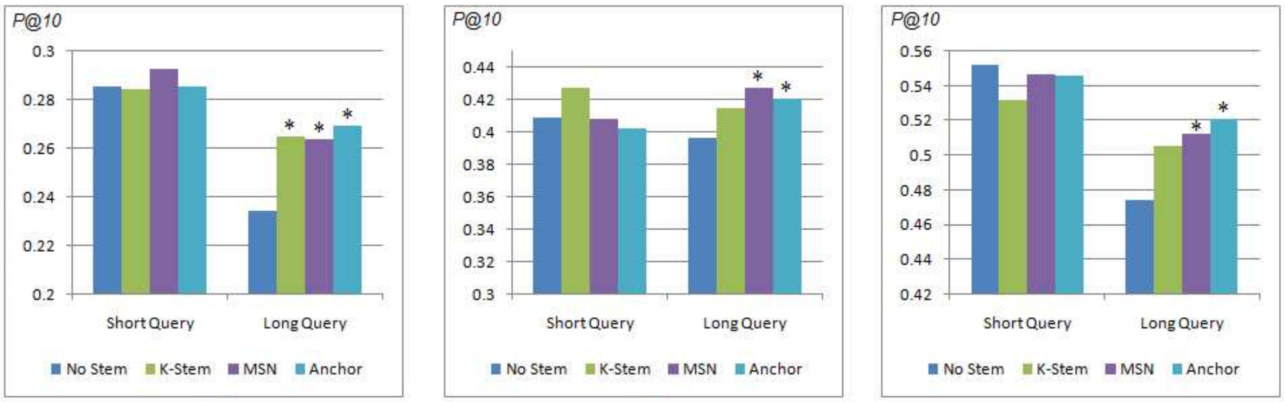
At learning time, we stem both logs using the Krovetz stemmer and apply the method described in Section 3 on each log to build the corresponding substitution model. Note that we did not stem the logs in stemming experiments. At query time, for each query q, the top m candidates given by the learned substitution model were used to do retrieval on indexes stemmed also by a Krovetz stemmer. The best P@5 obtained by these m candidates is recorded as the P@5 of the substitution solution for q. We varied m from 1 to 10 in our experiment. Due to the coverage problem, both logs cannot reformulate all queries associated with each collection. Therefore, we created a subset of queries where the translation model trained from both logs can provide "related" words to at least one query word and compared the two logs based on these query sets instead of the original ones. Information about these sets is given in Table 3.

5.3.2 Query Substitution with Short Queries

From Fig. 2 we can see that the substitution method does not work well on the TREC collections though it has been shown to work with web queries [23]. This result, however, is not unexpected. By substituting the original query word with another, we run a risk of changing the users' original intent. From the experiment the authors [23] described where complete substitution helps, we can see that that the original queries have a P@5 value of only about .04. This indicates that many web queries in the query log they used (MSN) were badly formulated (many of them contain spelling errors). Since the performance of the original queries is quite bad, it is relatively safe to do complete substitution. TREC queries, on the other hand, have much better quality and they yield a fairly good initial result - P@5 of 32.91%, 47.86% and 56.91% on WT10G, Robust04 and Gov-2 respectively. It should be noted that translation model, even with session information enhancement, does not produce synonyms, in fact, it produces only paradigmatically related words. For example, words such as "women", "men", "children" are considered related, and "diamond", "gold", "necklace" and "watches" are also considered related. Since these related words do not necessarily have similar meanings, replacing one word with another runs a very high risk of hurting the retrieval performance. Table 4 gives some examples for cases where substitution hurts retrieval performance. Consider the query "diamond smuggling" for example. Since "watches" is related to "diamond" and "watches smuggling" appears more often than "diamond smuggling" in the log, "watches smuggling" will be rated as a good query. However, a substitution like this totally changes the intent of the original query. Thus it is easy to understand why the performance can decrease.

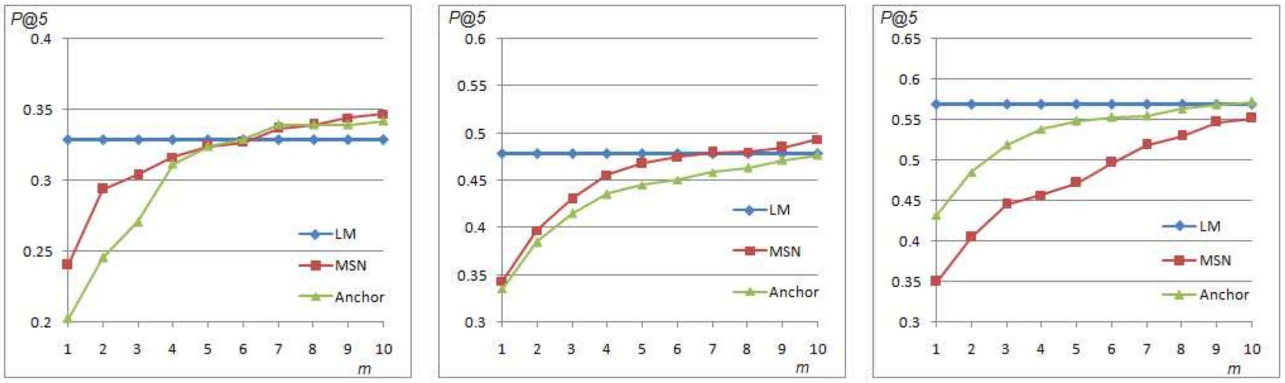
5.3.3 Query Expansion with Short Queries

Instead of substituting the original query word with the translation candidate, we can expand the original query with the candidate words. The candidate words and the original



(a) WT10G (b) Robust04 (c) Gov-2

Figure 1: Comparison in terms of P@10 among (i) context sensitive stemming, (ii) Krovetz stemming and (iii) no stemming with both short queries and long queries on three collections. We can see with long queries, context sensitive stemming is significantly ($p - value \leq 0.05$, marked by $*$) better than no stemming on all three collections while Krovetz is significantly better only on WT10G.

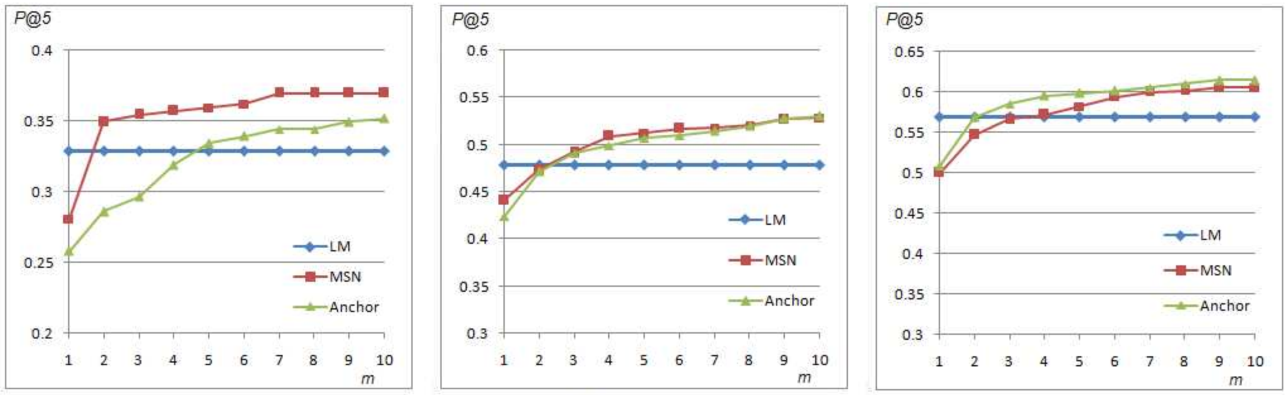


(a) WT10G (b) Robust04 (c) Gov-2

Figure 2: Query substitution results (P@5) with short queries on three collections

Table 4: Example of queries (for both MSN Log and anchor log) where substitution hurts the performance whereas expansion can retain or even improve the performance of the original query

	Original Query	P@5	Substituted Query	P@5	Expanded Query	P@5
MSN Log	diamond smuggling	1.0	watches smuggling	0.0	#syn(diamond watches) smuggling	1.0
	afghan women condition	0.8	afghan children condition	0.4	afghan #syn(women children) condition	1.0
	executive privilege	0.2	administrative privilege	0.0	#syn(executive administrative) privilege	0.2
	wetlands wastewater treatment	1.0	wetlands water treatment	0.8	wetlands #syn(wastewater water) treatment	1.0
	adult immigrants english	0.8	adult students english	0.6	adult #syn(immigrants students) english	0.8
	pearl farming	0.4	pearl planting	0.2	pearl #syn(farming planting)	0.4
Anchor Log	afghan women condition	0.8	afghan children condition	0.4	afghan #syn(women children) condition	1.0
	airport security	0.8	aviation security	0.6	#syn(airport aviation) security	0.8
	women parliaments	1.0	children parliaments	0.0	#syn(women children) parliaments	1.0
	secret shoppers	0.4	secret village	0.0	secret #syn(shoppers village)	0.4
	part time benefits	0.4	part time funds	0.2	part time #syn(benefits funds)	0.6
	heroic acts	0.2	heroic actions	0.0	heroic #syn(acts actions)	0.2



(a) WT10G (b) Robust04 (c) Gov-2

Figure 3: Query expansion results (P@5) with short queries on three collections

query term are combined using the #syn operator implemented in Indri.

As we can see from Fig. 3, the result obtained with expansion is much better than with substitution. On both Robust04 and Gov-2, query logs start to provide improvement over the original queries at $m = 3$. When we consider up to top 10 reformulations for each original query, the anchor log provides a relative improvement of 10.81% and 8% on Robust04 and Gov-2 respectively, and the MSN log gives a relative improvement of 10.44% and 6.46% on Robust04 and Gov-2 respectively. On WT10G, the MSN search log with expansion also improves the performance substantially, whereas the anchor log provides some improvement, but not as large.

Unlike substitution, expansion keeps the original words, thus does not completely change the original query intent. Table 4 shows that for those queries where substitution hurts performance, expansion can retain (or even improve in some case) the performance of the original query.

Table 5 gives some examples of good query expansions provided by the two logs. All these reformulated queries give better P@5 than the corresponding original queries. We can see that both logs can provide the same reformulation to some query. However, there are also queries where the two logs give different reformulation and those where one log can suggest good reformulations but the other does not.

5.3.4 *"Chance" versus "Risk" Analysis*

With both substitution and expansion, we add a new term into the original query. Obviously, once we do so, there is a chance the new word will bring some new relevant documents but we also run the risk of getting more non-relevant documents. Now we will look into the balance between "chance" and "risk" brought by the two methods. In order to do this, for each method we separate the original query set into two - (i) the subset of queries over which it provides improvement and (ii) the subset over which it hurts. We then compare the performance of the recommended queries and the original queries in each set. We use the term "Sub-Inc" and "Sub-Dec" to denote set (i) and (ii) of the substitution method and "Exp-Inc" and "Exp-Dec" to denote set (i) and (ii) of the expansion method respectively.

Table 6 shows that the relative improvement (on the set of improvable queries) given by the substitution technique is, in fact, a lot more than that given by the expansion technique. What this means is when substitution can help, it improves performance substantially, whereas the improvement given by expansion is not as large. However, the number of queries hurt by substitution is also considerably more than those hurt by expansion. Consequently, substitution overall cannot provide any significant improvement in P@5. Expansion, on the other hand, helps more than it hurts, thus resulting in much better overall performance.

We also note from Table 6 that the original queries that cannot be improved via reformulation (either substitution or expansion) have much better performance than those where reformulation can help. This confirms the intuition that it is safer to reformulate queries where initial performance is low. We believe that this partly explains why substitution helps in the case of web queries as observed by Wang and Zhai [23]. In addition, this also points out the limitation of the current reformulation technique: although it can give good reformulation in some cases, it is not reliable enough for determining whether or not to reformulate. This is why at $m = 1$, which means we consider only the best recommended query, both substitution and expansion hurts P@5 dramatically. More error analysis will be provided in the next section.

In summary, with the current reformulation techniques, despite the fact that completely substituting the original query term with a translation candidate helps in the case of web queries [23], it does not help on TREC collections. This is because TREC queries are reasonably well formulated, and the risk of getting more non-relevant documents introduced by the substituted queries is comparable or even higher than the benefit they bring. Expansion, on the other hand, works much better mainly because it can improve more queries than it hurts. Our results also show that an anchor log gives comparable performance to a real query log for this task.

5.3.5 *Error Analysis*

From Fig. 2 and Fig. 3 we can see that if we do either substitution or expansion using the best suggested query, the performance suffers badly. There are two reasons that account for this. The first, and main, reason is that the probability estimate given by formula (6) is not reliable enough. Formula (6) is used as the means of estimating how good the

Table 5: Examples of good expansion

	Original Query	Expanded Query	Original Query	Expanded Query
MSN Log	hunting deaths	hunting #syn(deaths accidents)	railway accidents	#syn(railway train) accidents
	new fuel sources	new #syn(fuel energy) sources	oscar winner selection	oscar winner #syn(selection promotion)
	educational standards	#syn(educational teaching) standards	marine vegetation	marine #syn(vegetation plants)
	automobile recalls	#syn(automobile auto) recalls	overseas tobacco sales	overseas #syn(tobacco cigarettes) sales
	doctor assisted suicides	#syn(doctor physicians) assisted suicides	food drug laws	food drug #syn(laws act)
	cheese production	cheese #syn(production companies)	volkswagen mexico	#syn(volkswagen vw) mexico
	illegal immigrant wages	illegal immigrant #syn(wages working)	chevrolet trucks	#syn(chevrolet chevy) trucks
Anchor Log	hunting deaths	hunting #syn(deaths accidents)	railway accidents	#syn(railway railroad) accidents
	new fuel sources	new #syn(fuel energy) sources	pearl farming	pearl #syn(farming industry)
	educational standards	#syn(educational teaching) standards	eskimo history eskimo	#syn(history culture)
	automobile recalls	#syn(automobile auto)	international art crime	international art #syn(crime fraud)
	doctor assisted suicides	#syn(doctor physicians) assisted suicides	wildlife extinction	#syn(wildlife animals) extinction
	cheese production	cheese #syn(production prices)	blood alcohol fatalities	blood alcohol #syn(fatalities deaths)
	illegal immigrant wages	illegal #syn(immigrant workers) wages	windmill electricity	windmill #syn(electricity power)

Table 6: Performance separated by "chance" and "risk". The 2nd column denotes the number of queries in each subset given in the 1st column. The 3rd column indicates P@5 of original queries in each subset. The 4th column indicates P@5 obtained by reformulated queries and the relative change to that of the original queries. "Affected queries" means the number of queries for which the model actually determines to generate substitution/expansion

Subset	# Query	Org. Q P@5	Reform. Q P@5
MSN Log			
WT10G (57 affected queries)			
Sub-Inc	18	0.2667	0.5778 (+116.65%)
Sub-Dec	14	0.4664	0.1429 (-69.36%)
Exp-Inc	16	0.3125	0.5625 (+80%)
Exp-Dec	5	0.3858	0.16 (-58.53%)
Robust04 (180 affected queries)			
Sub-Inc	63	0.3143	0.6032 (+91.92%)
Sub-Dec	42	0.6495	0.2857 (-56.01%)
Exp-Inc	55	0.3782	0.6291 (+66.34%)
Exp-Dec	10	0.5679	0.26 (-54.22%)
Gov-2 (112 affected queries)			
Sub-Inc	34	0.3471	0.6412 (+84.73%)
Sub-Dec	36	0.688	0.3278 (-52.35%)
Exp-Inc	27	0.4	0.6815 (+70.37%)
Exp-Dec	9	0.5743	0.2222 (-61.31%)
Anchor Log			
WT10G (57 affected queries)			
Sub-Inc	19	0.3368	0.6421 (+90.65%)
Sub-Dec	14	0.4378	0.0714 (-83.69%)
Exp-Inc	15	0.3467	0.6 (+73.06%)
Exp-Dec	6	0.5548	0.1667 (-69.95%)
Robust04 (176 affected queries)			
Sub-Inc	62	0.3387	0.629 (+85.71%)
Sub-Dec	53	0.6619	0.3057 (-53.81%)
Exp-Inc	58	0.3241	0.5793 (+78.74%)
Exp-Dec	10	0.6879	0.32 (-53.48%)
Gov-2 (99 affected queries)			
Sub-Inc	34	0.3235	0.6824 (+110.94%)
Sub-Dec	32	0.699	0.3125 (-55.29%)
Exp-Inc	32	0.3875	0.7313 (+88.72%)
Exp-Dec	14	0.7121	0.3286 (-53.85%)

Table 7: Examples of bad reformulations due to the unreliability of formula (6) and the log sparsity

Type	Query Context	Reformulation
Formula (6)	_ cancer treatments	prostate → breast
	_ cancer treatments	prostate → lung
	best retirement _	country → state
	_ security	airport → museum
	_ school success	magnet → charter
Sparsity	pearl _	farming → planting
	_ world war ii	portugal → soccer
	controlling acid _	rain → plant
	_ edwards womens issues	john → james

reformulation is, while what it really measures is how often we see a query in the log. The fact that a new query appears more often than the original one does not necessarily imply that the reformulation is a good one. Take the query "prostate cancer treatment" for example. Since "breast cancer", "lung cancer" and "prostate cancer" are different types of cancer, they tend to appear in the same context. Thus, it is reasonable that KL divergence extracts them as related words. It happens that "breast cancer treatment" and "lung cancer treatment" have higher frequencies in the logs, so the substitution model determines that "prostate" can be replaced with "breast" or "lung". It is obvious that such reformulations will lead to a performance drop since it completely changes the query intent.

The second reason is the sparsity of the log. For example, with "john edwards womens issues", the probability of seeing "john edwards womens" is too small. Thus, the substitution model decided to replace/expand "john" with other names such as "james". Table 7 shows more examples of poor reformulations caused by both problems.

5.3.6 *Query Reformulation with Long Queries*

Fig. 4 and Fig. 5 show the results for substitution and expansion of long queries respectively. In the case of expansion, the results are similar to those obtained with short queries. Substitution, on the other hand, is more effective in this case. It is comparable to expansion on WT10G and even slightly better than expansion on Robust04 and Gov-2.

Substitution, in fact, is a set of two actions: (i) dropping the original query term and (ii) adding the new term to the query. In order to study why substitution can be helpful on long queries, we examine the effect of (i) and (ii) separately.

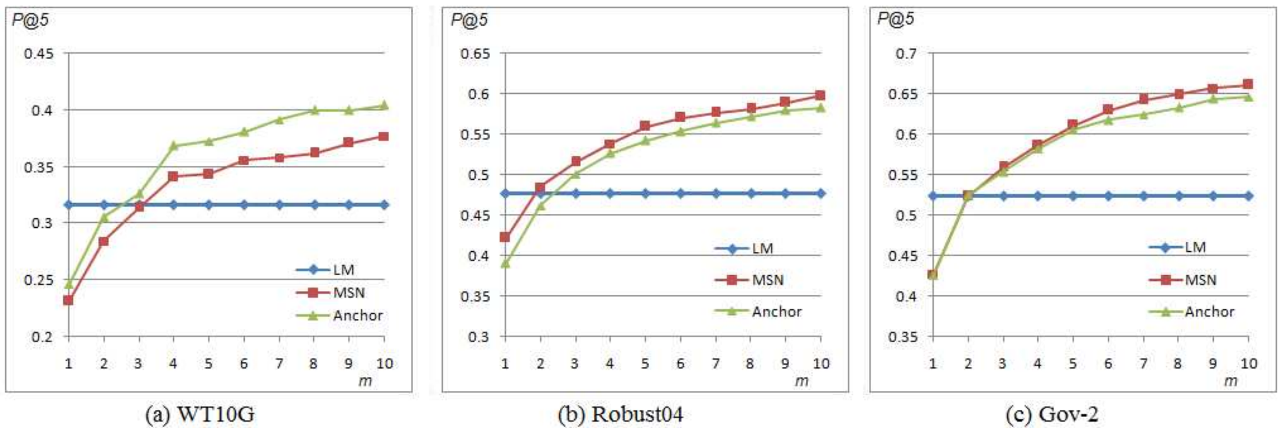


Figure 4: Query substitution results (P@5) with long queries on three collections

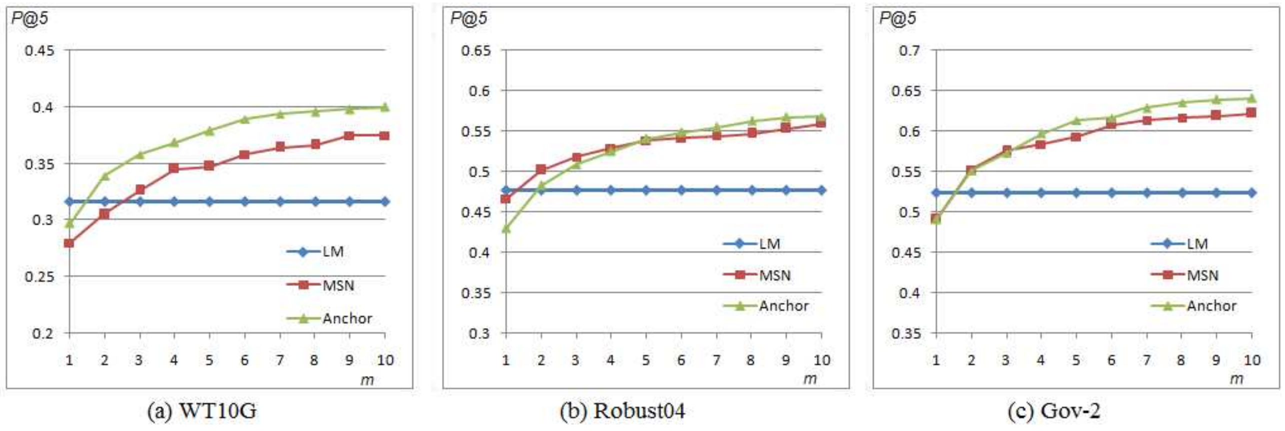


Figure 5: Query expansion results (P@5) with long queries on three collections

The experimental design is as follows. For each original query $q_{org} = w_1 \ldots w_{i-1} w_i w_{i+1} \ldots w_n$, we generate as before a list of substituted queries so that each new query is different to q_{org} at only one term and manually pick the one with highest P@5 as the substitution solution $q_{add} = w_1 \ldots w_{i-1} w_j w_{i+1} \ldots w_n$ for q_{org} where w_j is the substitution for w_i. Next, we create a second version of q_{add} in which we drop w_i, denoted as q_{drop}. We compare the performance of the original queries with the set of q_{add} and q_{drop}. This is to investigate how dropping the original query term and adding the new term separately affect retrieval performance.

Table 8 shows that in doing substitution with either log, in most cases, the improvement from Q_{drop} to Q_{add} is not very different on short and long queries. It is the change from the original query to Q_{drop} that makes the difference. With short queries, dropping a term always hurts performance and the adding of the new term increases the performance back up to the original level, resulting in no or very slight improvement overall. With long queries, dropping the original term actually increases the performance and then is further increased by adding the new term, resulting in a performance boost as seen in Fig. 4.

So, the fact that substitution works very well with long queries is not necessarily because it can suggest better substitution candidates for long queries but rather because long queries contain many redundant words, the elimination of which is beneficial. This is consistent with current research on long queries which states that either downweighting bad terms [4, 17] or dropping them [15] tends to help with long queries.

6. RELATED WORK

Query reformulation has a very long history, dating back to the earliest relevance feedback techniques [22] where queries are modified based on documents judged to be relevant or non-relevant. When explicit judgments are not available, pseudo-relevance feedback [26, 16] assumes top returned documents are relevant. More recent query reformulation techniques have exploited external sources such as query logs [23, 13, 3, 25, 27] and anchor text [19, 14]. In this paper, we look specifically at the technique proposed by Wang and Zhai [23], the most recent work in query reformulation using a query log. The authors have shown that the method is very helpful on web queries. We evaluate it on standard TREC collections and find that complete substitution methods are not very useful. Expansion, instead, produces much better results overall.

In this paper, we also slightly modify the substitution framework to do context sensitive query stemming, which results in a very similar approach to Peng et al. [20]. However, while they investigates specifically the case of pluraliza-

Table 8: Separated effects of dropping a term and adding a new term to the original query

			Q_{org}	Q_{drop}	Q_{add}
MSN Log	Short Q.	WT10G	0.3291	0.2734	0.3468
		Robust04	0.4786	0.4009	0.4937
		Gov-2	0.5632	0.4529	0.5515
	Long Q.	WT10G	0.3158	0.3074	0.3768
		Robust04	0.4764	0.5138	0.5976
		Gov-2	0.5238	0.5578	0.6612
Anchor Log	Short Q.	WT10G	0.3291	0.2962	0.3418
		Robust04	0.4786	0.3732	0.4768
		Gov-2	0.5632	0.4706	0.5721
	Long Q.	WT10G	0.3158	0.3432	0.4042
		Robust04	0.4764	0.513	0.5829
		Gov-2	0.5469	0.5361	0.6463

tion stemming, we study general stemming. In addition, we evaluate the technique on standard TREC collections and compare its performance to that of the conventional stemming approach.

In addition to query reformulation, query logs have been used for a variety of tasks. They have been used to learn retrieval functions [12, 1], spelling correction [2, 9], query segmentation [5] and disambiguating abbreviations [24]. In this paper, we studied reformulation tasks that can make use of the limited data in an anchor log, but some of these other log-based tasks may also be able to be tackled using an anchor log.

Carman et al. [7] compare query logs to the tags that users use to bookmark a web page. This work is very similar to ours in spirit but they are quite different in practice. Tags are not the same as anchor text and [7] focuses mainly on data analysis rather than retrieval experiments.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we evaluated two query reformulation methods using TREC collections and an anchor log and a query log. One of the main results of the paper is that an anchor log extracted from a web collection can be very competitive with a real query log for both query stemming and substitution. This extends the findings of existing research [11] which shows that anchor texts are close to real web queries in terms of term distribution and length, and shows the possibility of doing log-based research without proprietary data.

Our results with the most recent log-based query substitution method [23] show that it does not work very well with short queries on TREC collections because the quality of the initial query is quite good. Substituting query terms thus runs a high risk of changing the query intent. Expansion, on the other hand, is more reliable since it does not throw away the original words and provides significant overall improvement.

With long queries, while expansion is still effective, substitution is much more helpful than it is with short queries. The main reason for this is that long queries contain many words that are not needed for retrieval. Dropping these words (as part of substitution) improves the performance. Substitution consists of dropping the original term and adding a new term.

We also show that context sensitive stemming has considerable potential for improving effectiveness. In particular, it is consistently better than the conventional Krovetz stemming on long queries. It should be noted that the context sensitive stemming technique we used is very similar to expansion. This also helps to confirm that expansion is a very good method for reformulation.

One of the issues we observed was that, while the first recommended query given by the substitution rankings is often not effective, there is usually a very good reformulation somewhere in the top 10 list. We intend to apply machine learning techniques to rerank the list of recommended queries.

The fact that complete substitution helps with one kind of query and expansion helps with other leads to an interesting question: why have we been treating substitution and expansion separately? We believe that it will be important to develop a unified reformulation framework where the model can decide not only whether to reformulate query, but also whether to substitute or expand. We plan to investigate this in the future.

8. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #IIS-0711348, in part by NSF CLUE IIS-0844226, and in part by NSF grant #IIS-0534383. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

9. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of SIGIR*, pages 19-26, 2006.

[2] F. Ahmad and G. Kondrak. Learning a spelling error model from search query logs. In *Proceedings of HLT*, pages 955-962, 2005.

[3] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of KDD*, pages 407-416, 2000.

[4] M. Bendersky and W.B. Croft. Discovering key concepts in verbose queries. In *Proceedings of SIGIR*, pages 491-498, 2008.

[5] S. Bergsma and Q. Wang. Learning Noun Phrase Query Segmentation. In *Proceedings of EMNLP-CoNLL*, pages 819-826, 2007.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107-117, 1998.

[7] M. Carman, M. Baillie, R. Gwadera and F. Crestani. A statistical comparison of tag and query logs. In *Proceedings of SIGIR*, pages 123-130, 2009.

[8] W.B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice.* Addison-Wesley, 2009.

[9] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP*, pages 293-300, 2004.

[10] I. Dagan, F. Pereira and L. Lee. Similarity-Based Estimation of Word Cooccurrence Probabilities. In *Proceedings of ACL*, pages 272-278, 1994.

[11] N. Eiron and K.S. McCurley. Analysis of anchor text for web search. In *Proceedings of SIGIR*, pages 459-460, 2003.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of KDD*, pages 133-142, 2002.

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

[14] R. Kraft and J. Zien. Mining anchor text for query refinement. In *Proceedings of WWW*, pages 666-674, 2004.

[15] G. Kumaran and V.R. Carvalho. Reducing long queries using query quality predictors. In *Proceedings of SIGIR*, pages 564-571, 2009.

[16] V. Lavrenko and W.B. Croft. Relevance based language models. In *Proceedings of SIGIR*, pages 120-127, 2001.

[17] M. Lease, J. Allan and W.B. Croft. Regression Rank: Learning to Meet the Opportunity of Descriptive Queries. In *Proceedings of ECIR*, pages 90-101, 2009.

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

[19] R. Nallapati, W.B. Croft and J. Allan. Relevant query feedback in statistical language modeling. In *Proceedings of CIKM*, pages 560-563, 2003.

[20] F. Peng, N. Ahmed, X. Li, and Y. Lu. Context sensitive stemming for web search. In *Proceedings of SIGIR*, pages 639-646, 2007.

[21] F. Pereira, N. Tishby and L. Lee. Distributional Clustering of English Words. In *Proceedings of ACL*, pages 183-190, 1993.

[22] J. J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System Experiments in Automatic Document Processing*, pages 313-323, 1971.

[23] X. Wang and C. Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of CIKM*, pages 479-488, 2008.

[24] X. Wei, F. Peng, and B. Dumoulin. Analyzing web text association to disambiguate abbreviation in queries. In *Proceedings of SIGIR*, pages 751-752, 2008.

[25] J. Wen, J. Nie and H. Zhang. Clustering user queries of a search engine. In *Proceedings of WWW*, pages 162-168, 2001.

[26] J. Xu and W.B. Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inf. Syst.*, 18(1):79-112, 2000.

[27] R.B. Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *Proceedings of EDBT Workshop*, pages 588-596, 2004.