# Selecting Query Term Alterations for Web Search by Exploiting Query Contexts

**Guihong Cao**
Dept. of Computer Science and Operations Research
University of Montreal, Canada
caogui@iro.umontreal.ca

**Stephen Robertson**
Microsoft Research at Cambridge
Cambridge, UK
ser@microsoft.com

**Jian-Yun Nie**
Dept. of Computer Science and Operations Research
University of Montreal, Canada
nie@iro.umontreal.ca

## Abstract

Query expansion by word alterations (alternative forms of a word) is often used in Web search to replace word stemming. This allows users to specify particular word forms in a query. However, if many alterations are added, query traffic will be greatly increased. In this paper, we propose methods to select only a few useful word alterations for query expansion. The selection is made according to the appropriateness of the alteration to the query context (using a bigram language model), or according to its expected impact on the retrieval effectiveness (using a regression model). Our experiments on two TREC collections will show that both methods only select a few expansion terms, but the retrieval effectiveness can be improved significantly.

## 1 Introduction

Word stemming is a basic NLP technique used in most of Information Retrieval (IR) systems. It transforms words into their root forms so as to increase the chance to match similar words/terms that are morphological variants. For example, with stemming, "controlling" can match "controlled" because both have the same root "control". Most stemmers, such as the Porter stemmer (Porter, 1980) and Krovetz stemmer (Krovetz, 1993), deal with stemming by stripping word suffixes according to a set of morphological rules. Rule-based approaches are intuitive and easy to implement. However, while in general, most words can be stemmed correctly; there is often erroneous stemming that unifies unrelated words. For instance, "jobs" is stemmed to "job" in both "find jobs in Apple" and "Steve Jobs at Apple". This is particularly problematic in Web search, where users often use special or new words in their queries. A standard stemmer such as Porter's will wrongly stem them.

To better determine stemming rules, Xu and Croft (1998) propose a selective stemming method based on corpus analysis. They refine the Porter stemmer by means of word clustering: words are first clustered according to their co-occurrences in the text collection. Only word variants belonging to the same cluster will be conflated.

Despite this improvement, the basic idea of word stemming is to transform words in both documents and queries to a standard form. Once this is done, there is no means for users to require a specific word form in a query – the word form will be automatically transformed, otherwise, it will not match documents. This approach does not seem to be appropriate for Web search, where users often specify particular word forms in their queries. An example of this is a quoted query such as "Steve Jobs", or "US Policy". If documents are stemmed, many pages about job offerings or US police may be returned ("policy" conflates with "police" in Porter stemmer). Another drawback of stemming is that it usually enhances recall, but may hurt precision (Kraaij and Pohlmann, 1996). However, general Web search is basically a precision-oriented task.

One alternative approach to word stemming is to do query expansion at query time. The original query terms are expanded by their related forms having the same root. All expansions can be combined by the Boolean operator "*OR*". For example, the query "*controlling acid rain*" can be expanded to "*(control OR controlling OR controller OR controlled OR controls) (acid OR acidic OR acidify) (rain OR raining OR rained OR rains)*". We will call each such expansion term an *alteration* to the original query term. Once a set of possible alterations is determined, the simplest approach to perform expansion is to add all possible alterations. We call this approach ***Naive Expansion***. One can easily show that stemming at indexing time is equivalent to *Naive Expansion* at retrieval time. This approach has been adopted by most commercial search engines (Peng et al., 2007). However, the expansion approaches proposed previously can have several serious problems: First, they usually do not consider expansion ambiguity – each query term is usually expanded independently. However, some expansion terms may not be appropriate. The case of "Steve Jobs" is one such example, for which the word "job" can be proposed as an expansion term. Second, as each query term may have several alterations, the naïve approach using all the alterations will create a very long query. As a consequence, query traffic (the time required for the evaluation of a query) is greatly increased. Query traffic is a critical problem, as each search engine serves millions of users at the same time. It is important to limit the query traffic as much as possible.

In practice, we can observe that some word alterations are irrelevant and undesirable (as in the "Steve Jobs" case), and some other alterations have little impact on the retrieval effectiveness (for example, if we expand a word by a rarely used word form). In this study, we will address these two problems. Our goal is to select only appropriate word alterations to be used in query expansion. This is done for two purposes: On the one hand, we want to limit query traffic as much as possible when query expansion is performed. On the other hand, we also want to remove irrelevant expansion terms so that fewer irrelevant documents will be retrieved, thereby improve the retrieval effectiveness.

To deal with the two problems we mentioned above, we will propose two methods to select alterations. In the first method, we make use of the query context to select only the alterations that fit the query. The query context is modeled by a bigram language model. To reduce query traffic, we select only one alteration for each query term, which is the most coherent with the bigram model. We call this model ***Bigram Expansion***. Despite the fact that this method adds far fewer expansion terms than the naïve expansion, our experiments will show that we can achieve comparable or even better retrieval effectiveness.

Both the Naive Expansion and the Bigram Expansion determine word alterations solely according to general knowledge about the language (bigram model or morphological rules), and no consideration about the possible effect of the expansion term is made. In practice, some alterations will have virtually no impact on retrieval effectiveness. They can be ignored. Therefore, in our second method, we will try to predict whether an alteration will have some positive impact on retrieval effectiveness. Only the alterations with positive impact will be retained. In this paper, we will use a regression model to predict the impact on retrieval effectiveness. Compared to the bigram expansion method, the regression method results in even fewer alterations, but experiments show that the retrieval effectiveness is even better.

Experiments will be conducted on two TREC collections, Gov2 data for Web Track and TREC6&7&8 for ad-hoc retrieval. The results show that the two methods we propose both outperform the original queries significantly with less than two alterations per query on average. Compared to the *Naive Expansion* method, the two methods can perform at least equally well, while query traffic is dramatically reduced.

In the following section, we provide a brief review of related work. Section 3 shows how to generate alteration candidates using a similar approach to Xu and Croft's corpus analysis (1998). In section 4 and 5, we describe the Bigram Expansion method and Regression method respectively. Section 6 presents some experiments on TREC benchmarks to evaluate our methods. Section 7 concludes this paper and suggests some avenues for future work.

## 2 Related Work

Many stemmers have been implemented and used as standard processing in IR. Among them, the Porter stemmer (Porter, 1980) is the most widely used. It strips term suffixes step-by-step according to a set of morphological rules. However, the Porter stemmer sometimes wrongly transforms a term into an unrelated root. For example, it will unify "news" and "new", "execute" and "executive". On the other hand, it may miss some conflations, such as "mice" and "mouse", "europe" and "european". Krovetz (1993) developed another stemmer, which uses a machine-readable dictionary, to improve the Porter stemmer. It avoids some of the Porter stemmer's wrong stripping, but does not produce consistent improvement in IR experiments.

Both stemmers use generic rules for English to strip each word in isolation. In practice, the required stemming may vary from one text collection to another. Therefore, attempts have been made to use corpus analysis to improve existing rule-based stemmers. Xu and Croft (1998) create equivalence clusters of words which are morphologically similar and occur in similar contexts.

As we stated earlier, the stemming-based IR approaches are not well suited to Web search. Query expansion has been used as an alternative (Peng et al. 2007). To limit the number of expansion terms, and thus the query traffic, Peng et al. only use alterations for some of the query words: They segment each query into phrases and only the head word in each phrase is expanded. The assumptions are: 1)Queries issued in Web search often consist of noun phrases. 2) Only the head word in the noun phrase varies in form and needs to be expanded. However, both assumptions may be questionable. Their experiments did not show that the two assumptions hold.

Stemming is related to query expansion or query reformulation (Jones et al., 2006; Anick, 2003; Xu and Croft, 1996), although the latter is not limited to word variants. If the expansion terms used are those that are variant forms of a word, then query expansion can produce the same effect as word stemming. However, if we add all possible word alterations, query expansion/reformulation will run the risk of adding many unrelated terms to the original query, which may result in both heavy traffic and topic drift. Therefore, we need a way to select the most appropriate expansion terms. In (Peng et al. 2007), a bigram language model is used to determine the alteration of the head word that best fits the query. In this paper, one of the proposed methods will also use a bigram language model of the query to determine the appropriate alteration candidates. However, in our approach, alterations are not limited to head words. In addition, we will also propose a supervised learning method to predict if an alteration will have a positive impact on retrieval effectiveness. To our knowledge, no previous method uses the same approach.

In the following sections, we will describe our approach, which consists of two steps: the generation of alteration candidates, and the selection of appropriate alterations for a query. The first step is query-independent using corpus analysis, while the second step is query-dependent. The selected word alterations will be *OR*-ed with the original query words.

## 3 Generating Alteration Candidates

Our method to generate alteration candidates can be described as follows. First, we do word clustering using a Porter stemmer. All words in the vocabulary sharing the same root form are grouped together. Then we do corpus analysis to filter out the words which are clustered incorrectly, according to word distributional similarity, following (Xu and Croft, 1998; Lin 1998). The rationale behind this is that words sharing the same meaning tend to occur in the same contexts.

The context of each word in the vocabulary is represented by a vector containing the frequencies of the context words which co-occur with the word within a predefined window in a training corpus. The window size is set empirically at 3 words and the training corpus is about 1/10 of the GOV2 corpus (see section 5 for details about the collection). Similarity is measured by the cosine distance between two vectors. For each word, we select at most 5 similar words as alteration candidates.
In the next sections, we will further consider ways to select appropriate alterations according to the query.

## 4 Bigram Expansion Model for Alteration Selection

In this section, we try to select the most suitable alterations according to the query context. The query context is modeled by a bigram language model as in (Peng et al. 2007).

Given a query described by a sequence of words, we consider each of the query word as representing a concept $c_i$. In addition to the given word form, $c_i$ can also be expressed by other alternative forms. However, the appropriate alterations do not only depend on the original word of $c_i$, but also on other query words or their alterations.

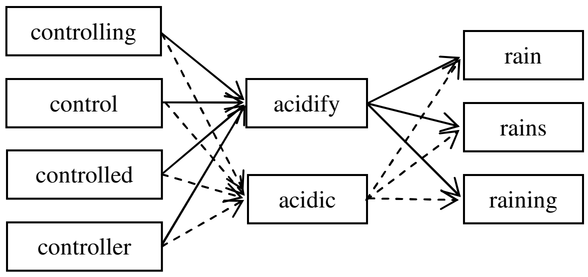

Figure 1: Considering all Combinations to Calculate the Plausibility of Alterations

Accordingly, a confidence weight is determined for each alteration candidate. For example, in the query "Steve Jobs at Apple", the alteration "job" of "jobs" should have a low confidence; while in the query "finding jobs in Apple", it should have a high confidence.

One way to measure the confidence of an alteration is the plausibility of its appearing in the query. Since each concept may be expressed by several alterations, we consider all the alterations of context concepts when calculating the plausibility of a given word. Suppose we have the query "controlling acid rain". The second concept has two alterations - "acidify" and "acidic". For each of the alterations, our method will consider all the combinations with other words, as illustrated in figure 1, where each combination is shown as a path. More precisely, for a query of $n$ words (or their corresponding concepts), let $e_{i,j} \in c_i$, $j=1,2,\ldots,|c_i|$ be the alterations of concept $c_i$. Then we have:

$$P(e_{ij}) = \sum_{1,j_1=1}^{|c_1|} \sum_{2,j_2=1}^{|c_2|} \cdots \sum_{i-1,j_{i-1}=1}^{|c_{i-1}|} \sum_{i+1,j_{i+1}=1}^{|c_{i+1}|} \cdots \\ \cdots \sum_{n,j_n=1}^{|c_n|} P(e_{1,j_1}, e_{2,j_2}, \ldots, e_{i,j_i}, \ldots, e_{n,j_n}) \quad (1)$$

In equation 1, $e_{1,j_1}, e_{2,j_2}, \ldots, e_{i,j_i}, \ldots, e_{n,j_n}$ is a path passing through $e_{i,j}$. For simplicity, we abbreviate it as $e_1 e_2 \ldots e_i \ldots e_n$. In this work, we used bigram language model to calculate the probability of each path. Then we have:

$$P(e_1, e_2, \ldots, e_i, \ldots, e_n) = P(e_1) \prod_{k=2}^{n} P(e_k \mid e_{k-1}) \quad (2)$$

$P(e_k|e_{k-1})$ is estimated with a back-off bigram language model (Goodman, 2001). In the experiments with TREC6&7&8, we train the model with all text collections; while in the experiments with Gov2 data, we only used about 1/10 of the GOV2 data to train the bigram model because the whole Gov2 collection is too large.

Directly calculating $P(e_{ij})$ by summing the probabilities of all paths passing through $e_{ij}$ is an NP problem (Rabiner, 1989), and is intractable if the query is long. Therefore, we use the forward-backward algorithm (Bishop, 2006) to calculate $P(e_{ij})$ in a more efficient way. After calculating $P(e_{ij})$ for each $c_i$, we select one alteration which has the highest probability. We limit the number of additional alterations to 1 in order to limit query traffic. Our experiments will show that this is often sufficient.

## 5 Regression Model for Alteration Selection

None of the previous selection methods considers how well an alteration would perform in retrieval. The Bigram Expansion model assumes that the query replaced with better alterations should have a higher likelihood. This approach belongs to the family of unsupervised learning. In this section, we introduce a method belonging to supervised learning family. This method develops a regression model from a set of training data, and it is capable of predicting the expected change in performance when the original query is augmented by this alteration. The performance change is measured by the difference in the Mean Average Precision (MAP) between the augmented and the original query. The training instances are defined by the original query string, an original query term under consideration and one alteration to the query term. A set of features will be used, which will be defined later in this section.

### 5.1 Linear Regression Model

The goal of the regression model is to predict the performance change when a query term is augmented with an alteration. There are several regression models, ranging from the simplest linear regression model to non-linear alternatives, such as a neural network (Duda et al., 2001), a Regression SVM (Bishop, 2006). For simplicity, we use linear regression model here. We denote an instance in the feature space as $X$, and the weights of features are denoted as $W$. Then the linear regression model is defined as:

$$f(X) = W^T X \quad (3)$$

where $W^T$ is the transpose of $W$. However, we will have a technical problem if we set the target value to the performance change directly: The range of values of *f(X)* is $(-\infty,+\infty)$, while the range of performance change is [-1,1]. The two value ranges do not match. This inconsistency may result in severe problems when the scales of feature values vary dramatically (Duda et al., 2001). To solve this problem, we do a simple transformation on the performance change. Let the change be $y \in [-1,1]$, then the transformed performance change is:

$$\varphi(y) = \log \frac{1+y+\gamma}{1-y+\gamma} \quad y \in [-1,1] \tag{4}$$

where $\gamma$ is a very small positive real number (set to be 1*e*-37 in the experiments), which acts as a smoothing factor. The value of $\varphi(y)$ can be an arbitrary real number. $\varphi(y)$ is a monotonic function defined in the range of [-1,1]. Moreover, the fixed point of $\varphi(y)$ is 0, i.e., $\varphi(y)=y$ when y=*0*. This property is nice; it means that the expansion brings positive improvement if and only if *f(X)>0,* which makes it easy to determine which alteration is better.

We train the regression model by minimizing the mean square error. Suppose there are training instances $X_1,X_2,\ldots,X_m$, and the corresponding performance change is $y_i$, $i=1,2,\ldots,m$. We calculate the mean square error with the following equation:

$$err(W) = \sum_{i=1}^{m} (W^T X_i - \varphi(y_i))^2 \tag{5}$$

Then the optimal weight is defined as:

$$\begin{aligned} W^* &= \arg\min_W err(W) \\ &= \arg\min_W \sum_{i=1}^{m} (W^T X_i - \varphi(y_i))^2 \end{aligned} \tag{6}$$

Because *err(W)* is a convex function of *W*, it has a global minimum and obtains its minimum when the gradient is zero (Bazaraa et al., 2006). Then we have:

$$\frac{\partial err(W^*)}{\partial W^*} = \sum_{i=1}^{m} (W^T X_i - \varphi(y_i)) X_i^T = 0$$

So, $W^{*T} \sum_{i=1}^{m} X_i X_i^T = \sum_{i=1}^{m} \varphi(y_i) X_i^T$

In fact, $\sum_{i=1}^{m} X_i X_i^T$ is a square matrix, we denote it as $XX^T$. Then we have:

$$W^* = (XX^T)^{-1} \left[ \sum_{i=1}^{m} \varphi(y_i) X_i \right] \tag{7}$$

The matrix $XX^T$ is an $l \times l$ square matrix, where *l* is the number of features. In our experiments, we only use three features. Therefore the optimal weights can be calculated efficiently even we have a large number of training instances.

## 5.2 Constructing Training Data

As a supervised learning method, the regression model is trained with a set of training data. We illustrate here the procedure to generate training instances with an example.

Given a query "controlling acid rain", we obtain the MAP of the original query at first. Then we augment the query with an alteration to the original term (one term at a time) at each time. We retain the MAP of the augmented query and compare it with the original query to obtain the performance change. For this query, we expand "controlling" by "control" and get an augmented query "*(controlling OR control) acid rain*". We can obtain the difference between the MAP of the augmented query and that of the original query. By doing this, we can generate a series of training instances consisting of the original query string, the original query term under consideration, its alteration and the performance change, for example:

*<controlling acid rain, controlling, control, 0.05>*

Note that we use MAP to measure performance, but we could well use other metrics such as NDCG (Peng et al., 2007) or P@N (precision at top-N documents).

## 5.3 Features Used for Regression Model

Three features are used. The first feature reflects to what degree an alteration is coherent with the other terms. For example, for the query "controlling acid rain", the coherence of the alteration "acidic" is measured by the logarithm of its co-occurrence with the other query terms within a predefined window (90 words) in the corpus. That is:

*log(count(controlling…acidic…rain|window)+0.5)*

where "…" means there may be some words between two query terms. Word order is ignored.

The second feature is an extension to point-wise mutual information (Rijsbergen, 1979), defined as follows:

$$\log\left(\frac{P(controlling...acidic...rain \mid window)}{P(controlling)P(acidic)P(rain)}\right)$$

where *P(controlling…acidic…rain|window)* is the co-occurrence probability of the trigram containing acidic within a predefined window (50 words). *P(controlling), p(acidic), P(rain)* are probabilities of the three words in the collection. The three words are defined as: the term under consideration, the first term to the left of that term, and the first term to the right. If a query contains less than 3 terms or the term under consideration is the beginning/ending term in the query, we will set the probability of the missed term/terms to be 1. Therefore, it becomes point-wise mutual information when the query contains only two terms. In fact, this feature is supplemental to the first feature. When the query is very long and the first feature always obtains a value of *log(0.5)*, so it does not have any discriminative ability. On the other hand, the second feature helps because it can capture some co-occurrence information no matter how long the query is.

The last feature is the bias, whose value is always set to be 1.0.

The regression model is trained in a leave-one-out cross-validation manner on three collections; each of them is used in turn as a test collection while the two others are used for training. For each incoming query, the regression model predicts the expected performance change when one alteration is used. For each query term, we only select the alteration with the largest positive performance change. If none of its alterations produce a positive performance change, we do not expand the query term. This selection is therefore more restrictive than the Bigram Expansion Model. Nevertheless, our experiments show that it improves retrieval effectiveness further.

## 6 Experiments

### 6.1 Experiment Settings

In this section, our aim is to evaluate the two context-sensitive word alteration selection methods. The ideal evaluation corpus should be composed of some Web data. Unfortunately, such data are not publicly available and the results also could not be compared with other published results. Therefore, we use two TREC collections. The first one is the ad-hoc retrieval test collections used for TREC6&7& 8. This collection is relative small and homogeneous. The second one is the Gov2 data. It is obtained by crawling the entire .gov domain and has been used for three TREC Terabyte tracks (TREC2004-2006). Table 1 shows some statistics of the two collections. For each collection, we use 150 queries. Since the Regression model needs some data for training, we divided the queries into three parts, each containing 50 queries. We then use leave-one-out cross-validation. The evaluation metrics shown below are the average value of the three-fold cross-validation. Because the queries in Web are usually very short, we use only the title field of each query.

| **Name** | **Description** | **Size (GB)** | **#Doc** | **Query** |
|---|---|---|---|---|
| **TREC6 &7&8** | TREC disk4&5, Newpapers | 1.7 | 500,447 | 301-450 |
| **Gov2** | 2004 crawl of entire .gov domain | 427 | 25,205,179 | 701-850 |

Table1: Overview of Test Collections

To correspond to Web search practice, both documents and queries are not stemmed. We do not filter the stop words either.

Two main metrics are used: the Mean Average Precision (MAP) for the top 1000 documents to measure retrieval effectiveness, and the number of terms in the query to reflect query traffic. In addition, we also provide precision for the top 30 documents (P@30) to show the impact on top ranked documents. We also conducted t-tests to determine whether the improvement is statistically significant.

The Indri 2.5 search engine (Strohman et al., 2004) is used as our basic retrieval system. It provides for a rich query language allowing disjunctive combinations of words in queries.

### 6.2 Experimental Results

The first baseline method we compare with only uses the original query, which is named *Original*. In addition to this, we also compare with the following methods:

*Naïve Exp*: The Naïve expansion model expands each query term with all terms in the vocabulary sharing the same root with it. This model is equivalent to the traditional stemming method.

*UMASS*: This is the result reported in (Metzler et al., 2006) using Porter stemming for both document and query terms. This reflects a state-of-the-art result using Porter stemming.

*Similarity*: We select the alterations (at most 5) with the highest similarity to the original term. This is the method described in section 3.

The two methods we propose in this paper are the following ones:

*Bigram Exp*: the alteration is chosen by a Bigram Expansion model.

*Regression*: the alteration is chosen by a Regression model.

| Model | P@30 | #term | MAP | Imp. |
|---|---|---|---|---|
| **Original** | 0.4701 | 158 | 0.2440 | ---- |
| **UMASS** | ------- | ------- | 0.2666 | 9.26 |
| **Naïve Exp** | 0.4714 | 1345 | 0.2653 | 8.73 |
| **Similarity** | 0.4900 | 303 | 0.2689 | 10.20* |
| **Bigram Exp** | 0.5007 | 303 | 0.2751 | 12.75** |
| **Regression** | 0.5054 | 237 | 0.2773 | 13.65** |

Table 2: Results of Query 701-750 Over Gov2 Data

| Model | P@30 | #term | MAP | Imp. |
|---|---|---|---|---|
| **Original** | 0.4907 | 158 | 0.2738 | ---- |
| **UMASS** | ------- | ------- | 0.3251 | 18.73 |
| **Naive Exp** | 0.5213 | 1167 | 0.3224 | 17.75** |
| **Similarity** | 0.5140 | 290 | 0.3043 | 11.14** |
| **Bigram Exp.** | 0.5153 | 290 | 0.3107 | 13.47** |
| **Regression** | 0.5140 | 256 | 0.3144 | 14.82** |

Table 3: Results of Query 751-800 over Gov2 Data

| Model | P@30 | #term | MAP | Imp. |
|---|---|---|---|---|
| **Original** | 0.4710 | 154 | 0.2887 | ---- |
| **UMASS** | ------- | ------- | 0.2996 | 3.78 |
| **Naïve Exp** | 0.4633 | 1225 | 0.2999 | 3.87 |
| **Similarity** | 0.4710 | 288 | 0.2976 | 3.08 |
| **Bigram Exp** | 0.4730 | 288 | 0.3137 | 8.66** |
| **Regression** | 0.4748 | 237 | 0.3118 | 8.00* |

Table 4: Results of Query 801-850 over Gov2 Data

| Model | P@30 | #term | MAP | Imp. |
|---|---|---|---|---|
| **Original** | 0.2673 | 137 | 0.1669 | ---- |
| **Naïve Exp** | 0.3053 | 783 | 0.2146 | 28.57** |
| **Similarity** | 0.3007 | 255 | 0.2020 | 21.03** |
| **Bigram Exp** | 0.3033 | 255 | 0.2091 | 25.28** |
| **Regression** | 0.3113 | 224 | 0.2161 | 29.48** |

Table 5: Results of Query 301-350 over TREC6&7&8

| Model | P@30 | #term | MAP | Imp. |
|---|---|---|---|---|
| **Original** | 0.2820 | 126 | 0.1639 | ----- |
| **Naive Exp** | 0.2787 | 736 | 0.1665 | 1.59 |
| **Similarity** | 0.2867 | 244 | 0.1650 | 0.67 |
| **Bigram Exp.** | 0.2800 | 244 | 0.1641 | 0.12 |
| **Regression** | 0.2867 | 214 | 0.1664 | 1.53 |

Table 6: Results of Query 351-400 over TREC6&7&8

| Model | P@30 | #term | MAP | Imp. |
|---|---|---|---|---|
| **Original** | 0.2833 | 124 | 0.1759 | ----- |
| **Naïve Exp** | 0.3167 | 685 | 0.2138 | 21.55** |
| **Similarity** | 0.3080 | 240 | 0.2066 | 17.45** |
| **Bigram Exp** | 0.3133 | 240 | 0.2080 | 18.25** |
| **Regression** | 0.3220 | 187 | 0.2144 | 21.88** |

Table7: Results of Query 401-450 over TREC6&7&8

Tables 2, 3, 4 show the results of Gov2 data while table 5, 6, 7 show the results of the TREC6&7&8 collection. In the tables, the * mark indicates that the improvement over the original model is statistically significant with p-value<0.05, and ** means the p-values<0.01.

From the tables, we see that both word stemming (UMASS) and expansion with word alterations can improve MAP for all six tasks. In most cases (except in table 4 and 6), it also improve the precision of top ranked documents. This shows the usefulness of word stemming or word alteration expansion for IR.

We can make several additional observations:

1). Stemming Vs Expansion. UMASS uses document and query stemming while *Naive Exp* uses expansion by word alteration. We stated that both approaches are equivalent. The equivalence is confirmed by our experiment results: for all Gov2 collections, these approaches perform equivalently.

2). The Similarity model performs very well. Compared with the Naïve Expansion model, it produces quite similar retrieval effectiveness, while the query traffic is dramatically reduced. This approach is similar to the work of Xu and Croft (1998), and can be considered as another state-of-the-art result.

3). In comparison, the Bigram Expansion model performs better than the Similarity model. This shows that it is useful to consider query context in selecting word alterations.

4). The Regression model performs the best of all the models. Compared with the Original query, it adds fewer than 2 alterations for each query on average (since each group has 50 queries); nevertheless we obtained improvements on all the six collections. Moreover, the improvements on five collections are statistically significant. It also performs slightly better than the Similarity and Bigram Expansion methods, but with fewer alterations. This shows that the supervised learning approach, if used in the correct way, is superior to an unsupervised approach. Another advantage over the two other models is that the Regression model can reduce the number of alterations further. Because the Regression model selects alterations according to their expected improvement, the improvement of the alterations to one query term can be compared with that of the alterations to other query terms. Therefore, we can select at most one optimal alteration for the whole query. However, with the Similarity or Bigram Expansion models, the selection value, either similarity or query likelihood, cannot be compared across the query terms. As a consequence, more alterations need to be selected, leading to heavier query traffic.

## 7 Conclusion

Traditional IR approaches stem terms in both documents and queries. This approach is appropriate for general purpose IR, but is ill-suited for the specific retrieval needs in Web search such as quoted queries or queries with a specific word form that should not be stemmed. The current practice in Web search is not to stem words in index, but rather to perform a form of expansion using word alteration.

However, a naïve expansion will result in many alterations and this will increase the query traffic. This paper has proposed two alternative methods to select precise alterations by considering the query context. We seek to produce similar or better improvements in retrieval effectiveness, while limiting the query traffic.

In the first method proposed – the Bigram Expansion model, query context is modeled by a bigram language model. For each query term, the selected alteration is the one which maximizes the query likelihood. In the second method - Regression model, we fit a regression model to calculate the expected improvement when the original query is expanded by an alteration. Only the alteration that is expected to yield the largest improvement to retrieval effectiveness is added.

The proposed methods were evaluated on two TREC benchmarks: the ad-hoc retrieval test collection for TREC6&7&8 and the Gov2 data. Our experimental results show that both proposed methods perform significantly better than the original queries. Compared with traditional word stemming or the naïve expansion approach, our methods can not only improve retrieval effectiveness, but also greatly reduce the query traffic.

This work shows that query expansion with word alterations is a reasonable alternative to word stemming. It is possible to limit the query traffic by a query-dependent selection of word alterations. Our work shows that both unsupervised and supervised learning can be used to perform alteration selection.

Our methods can be further improved in several aspects. For example, we could integrate other features in the regression model, and use other nonlinear regression models, such as Bayesian regression models (e.g. Gaussian Process regression) (Rasmussen and Williams, 2006). The additional advantage of these models is that we can not only obtain the expected improvement in retrieval effectiveness for an alteration, but also the probability of obtaining an improvement (i.e. the robustness of the alteration).

Finally, it would be interesting to test the approaches using real Web data.

## References

Anick, P. (2003) Using Terminological Feedback for Web Search Refinement: a Log-based Study. In SIGIR, pp. 88-95.

Bazaraa, M., Sherali, H., and Shett, C. (2006). Nonlinear Programming, Theory and Algorithms. John Wiley & Sons Inc.

Bishop, C. (2006). Pattern Recognition and Machine Learning. Springer.

Duda, R., Hart, P., and Stork, D. (2001). Pattern Classification, John Wiley & Sons, Inc.

Goodman, J. (2001). A Bit of Progress in Language Modeling. Technical report.

Jones, R., Rey, B., Madani, O., and Greiner, W. (2006). Generating Query Substitutions. In WWW2006, pp. 387-396

Kraaij, W. and Pohlmann, R. (1996) Viewing Stemming as Recall Enhancement. Proc. SIGIR, pp. 40-48.

Krovetz, R. (1993). Viewing Morphology as an Inference Process. Proc. ACM SIGIR, pp. 191-202.

Lin, D. (1998). Automatic Retrieval and Clustering of Similar Words. In COLING-ACL, pp. 768-774.

Metzler, D., Strohman, T. and Croft, B. (2006). Indri TREC Notebook 2006: Lessons learned from Three Terabyte Tracks. In the Proceedings of TREC 2006.

Peng, F., Ahmed, N., Li, X., and Lu, Y. (2007). Context Sensitive Stemming for Web Search. Proc. ACM SIGIR, pp. 639-636 .

Porter, M. (1980) An Algorithm for Suffix Stripping. Program, 14(3): 130-137.

Rabiner, L. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In Proceedings of IEEE Vol. 77(2), pp. 257-286.

Rijsbergen, V. (1979). Information Retrieval. Butterworths, second version.

Strohman, T., Metzler, D. and Turtle, H., and Croft, B. (2004). Indri: A Language Model-based Search Engine for Complex Queries. In Proceedings of the International conference on Intelligence Analysis.

Xu, J. and Croft, B. (1996). Query Expansion Using Local and Global Document Analysis. Proc. ACM SIGIR, pp. 4-11.

Xu, J. and Croft, B. (1998). Corpus-based Stemming Using Co-occurrence of Word Variants. ACM TOIS, 16(1): 61-81.