Selecting Effective Terms for Query Formulation

Chia-Jung Lee, Yi-Chun Lin, Ruey-Cheng Chen, and Pu-Jen Cheng

Department of Computer Science and Information Engineering National Taiwan University
{r97037,r95021,pjcheng}@csie.ntu.edu.tw
cobain@turing.csie.ntu.edu.tw

Abstract. It is difficult for users to formulate appropriate queries for search. In this paper, we propose an approach to query term selection by measuring the effectiveness of a query term in IR systems based on its linguistic and statistical properties in document collections. Two query formulation algorithms are presented for improving IR performance. Experiments on NTCIR-4 and NTCIR-5 ad-hoc IR tasks demonstrate that the algorithms can significantly improve the retrieval performance by 9.2% averagely, compared to the performance of the original queries given in the benchmarks.

Keywords: Query Formulation, Query Term Selection.

1 Introduction

Users are often supposed to give effective queries so that the return of an information retrieval (IR) system is anticipated to cater to their information needs. One major challenge they face is what terms should be generated when formulating the queries. The general assumption of previous work [14] is that nouns or noun phrases are more informative than other parts of speech (POS), and longer queries could provide more information about the underlying information need. However, are the query terms that the users believe to be well-performing really effective in IR?

Consider the following description of the information need of a user, which is an example description query in NTCIR-4: *Find articles containing the reasons for NBA Star Michael Jordan's retirement and what effect it had on the Chicago Bulls*. Removing stop words is a common way to form a query such as "*contain*, *reason*, *NBA Star*, *Michael Jordan*, *retirement*, *effect*, *had*, *Chicago Bulls*", which scores a mean average precision (MAP) of 0.1914. It appears obviously that terms *contain* and *had* carry relatively less information about the topic. Thus, we take merely nouns into account and generate another query, "*reason*, *NBA Star*, *Michael Jordan*, *retirement*, *effect*, *Chicago Bulls"*, which achieves a better MAP of 0.2095. When carefully analyzing these terms, one could find that the meaning of *Michael Jordan* is more precise than that of *NBA Star*, and hence we improve MAP by 14% by removing *NBA Star*. Yet interestingly, the performance of removing *Michael Jordan* is not as worse as we think it would be. This might be resulted from that *Michael Jordan* is a famous *NBA*

G.G. Lee et al. (Eds.): AIRS 2009, LNCS 5839, pp. 168–180, 2009.

Star in *Chicago Bulls*. However, what if other terms such as *reason* and *effect* are excluded? There is no explicit clue to help users determine what terms are effective in an IR system, especially when they lack experience of searching documents in a specific domain. Without comprehensively understanding the document collection to be retrieved, it is difficult for users to generate appropriate queries.

As the effectiveness of a term in IR depends on not only how much information it carries in a query (subjectivity from users) but also what documents there are in a collection (objectivity from corpora), it is, therefore, important to measure the effectiveness of query terms in an automatic way. Such measurement is useful in selection of effective and ineffective query terms, which can benefit many IR applications such as query formulation and query expansion.

Conventional methods of retrieval models, query reformulation and expansion [13] attempt to learn a weight for each query term, which in some sense corresponds to the importance of the query term. Unfortunately, such methods could not explain what properties make a query term effective for search. Our work resembles some previous works with the aim of selecting effective terms. [1,3] focus on discovering key concepts from noun phrases in verbose queries with different weightings. Our work focuses on how to formulate appropriate queries by selecting effective terms or dropping ineffective ones. No weight assignments are needed and thus conventional retrieval models could be easily incorporated. [4] uses a supervised learning method for selecting good expansion terms from a number of candidate terms generated by pseudo-relevance feedback technique. However, we differ in that, (1) [4] selects specific features so as to emphasize more on the relation between original query and expansion terms without consideration of linguistic features, and (2) our approach does not introduce extra terms for query formulation. Similarly, [10] attempts to predict which words in query should be deleted based on query logs. Moreover, a number of works [2,5,6,7,9,15,16,18,19,20] pay attention to predict the quality or difficulty of queries, and [11,12] try to find optimal sub-queries by using maximum spanning tree with mutual information as the weight of each edge. However, their focus is to evaluate performance of *a* whole query whereas we consider units at the level of terms.

Given a set of possible query terms that a user may use to search documents relevant to a topic, the goal of this paper is to formulate appropriate queries by selecting effective terms from the set. Since exhaustively examining all candidate subsets is not feasible in a large scale, we reduce the problem to a simplified one that iteratively selects effective query terms from the set. We are interested in realizing (1) what characteristic of a query term makes it effective or ineffective in search, and (2) whether or not the effective query terms (if we are able to predict) can improve IR performance. We propose an approach to automatically measure the effectiveness of query terms in IR, wherein a regression model learned from training data is applied to conduct the prediction of term effectiveness of testing data. Based on the measurement, two algorithms are presented, which formulate queries by selecting effective terms and dropping ineffective terms from the given set, respectively.

The merit of our approach is that we consider various aspects that may influence retrieval performance, including linguistic properties of a query term and statistical relationships between terms in a document collection such as co-occurrence and

context dependency. Their impacts on IR have been carefully examined. Moreover, we have conducted extensive experiments on NTCIR-4 and NTCIR-5 ad-hoc IR tasks to evaluate the performance of the proposed approach. Based on term effectiveness prediction and two query formulation algorithms, our method significantly improve MAP by 9.2% on average, compared to the performance of the original queries given in the benchmarks.

In the rest of this paper, we describe the proposed approach to term selection and query formulation in Section 2. The experimental results of retrieval performance are presented in Sections 3. Finally, in Section 4, we give our discussion and conclusions.

2 Term Selection Approach for Query Formulation

2.1 Problem Specification

When a user desires to retrieve information from document repositories to know more about a topic, many possible terms may come into her mind to form various queries. We call such set of the possible terms *query term space* $T=\{t_1, \ldots, t_n\}$. A query typically consists of a subset of T. Each query term $t_i \in T$ is expected to convey some information about the user's information need. It is, therefore, reasonable to assume that each query term will have different degree of effectiveness in documents retrieval. Suppose Q denotes all subsets of T, that is, $Q=Power\ Set(T)$ and $|Q|=2^n$. The problem is to choose the best subset Δq^* among all candidates Q such that the performance gain between the retrieval performance of T and Δq ($\Delta q \in Q$) is maximized:

$$\Delta q^* = argmax_{\Delta q \in Q}\{(pf(T) - pf(\Delta q))/pf(T)\}. \quad (1)$$

where $pf(x)$ denotes a function measuring retrieval performance with x as the query. The higher the score $pf(x)$ is, the better the retrieval performance can be achieved.

An intuitive way to solve the problem is to exhaustively examine all candidate subset members in Q and design a method to decide which the best Δq^* is. However, since an exhaustive search is not appropriate for applications in a large scale, we reduce the problem to a simplified one that chooses the most effective query term t_i ($t_i \in T$) such that the performance gain between T and $T-\{t_i\}$ is maximized:

$$t_i^* = argmax_{t_i \in T}\{(pf(T) - pf(T - \{t_i\}))/pf(T)\}. \quad (2)$$

Once the best t_i^* is selected, Δq^* could be approximated by iteratively selecting effective terms from T. Similarly, the simplified problem could be to choose the most ineffective terms from T such that the performance gain is minimized. Then Δq^* will be approximated by iteratively removing ineffective or noisy terms from T.

Our goals are: (1) to find a function r: $T \rightarrow R$, which ranks $\{t_1, \ldots, t_n\}$ based on their effectiveness in performance gain (MAP is used for the performance measurement in this paper), where the effective terms are selected as candidate query terms, and (2) to formulate a query from the candidates selected by function r.

2.2 Effective Term Selection

To rank term t_i in a given query term space T based on function r, we use a regression model to compute r directly, which predicts a real value from some observed features of t_i. The regression function r: $T \rightarrow R$ is generated by learning from each t_i with the examples in form of $<f(t_i),(pf(T) - pf(T - \{t_i\}))/pf(T)>$ for all queries in the training corpus, where $f(t_i)$ is the feature vector of t_i, which will be described in Section 2.4.

The regression model we adopt is Support Vector Regression (SVR), which is a regression analysis technique based on SVM [17]. The aim of SVR is to find the most appropriate hyperplane **w** which is able to predict the distribution of data points accurately. Thus, r can be interpreted as a function that seeks the least dissimilarity between ground truth $y_i = (pf(T) - pf(T - \{t_i\}))/pf(T)$ and predicted value $r(t_i)$, and r is required to be in the form of $\boldsymbol{w}\, f(t_i)+b$. Finding function r is therefore equivalent to solving the convex optimization problem:

$$Min_{w,b,\xi_{i,1},\xi_{i,2}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i(\xi_{i,1} + \xi_{i,2}). \tag{3}$$

subject to:

$$\forall\, t_i \in T \qquad y_i \quad (\mathbf{w}\, f(t_i)+b) \geq \varepsilon + \xi_{i,1} \tag{4}$$

$$\forall\, i:\ \xi_{i,1}, \xi_{i,2} \geq 0 \qquad (\mathbf{w}\, f(t_i)+b) \quad y_i \geq \varepsilon + \xi_{i,2}\,. \tag{5}$$

where C determines the tradeoff between the flatness of r and the amount up to which deviations larger than ε are tolerated, ε is the maximum acceptable difference between the predicted and actual values we wish to maintain, and $\xi_{i,1}$ and $\xi_{i,2}$ are slack variables that cope with otherwise infeasible constraints of the optimization problem. We use the SVR implementation of LIBSVM [8] to solve the optimization problem.

Ranking terms in query term space $T=\{t_1, \ldots, t_n\}$ according to their effectiveness is then equivalent to applying regression function to each t_i; hence, we are able to sort terms $t_i \in T$ into an ordering sequence of effectiveness or ineffectiveness by $r(t_i)$.

2.3 Generation and Reduction

Algorithms *Generation* and *Reduction* formulate queries by greedily selecting effective terms or dropping ineffective terms from space T based on function r.

When formulating a query from query term space T, the Generation algorithm computes a measure of effectiveness $r(t_i)$ for each term $t_i \in T$, includes the most effective term t_i^* and repeats the process until k terms are chosen (where k is a empirical value given by users). Note that T is changed during the selection process, and thus statistical features should be re-estimated according to new T. The selection of the best candidate term ensures that the current selected term t_i^* is the most informative one among those that are not selected yet.

Compared to generation, the Reduction algorithm always selects the most ineffective term from current T in each iteration. Since users may introduce noisy terms in query term space T, Reduction aims to remove such ineffective terms and will repeat the process until $|T|$-k terms are chosen.

Algorithm Generation

Input: $T=\{t_1,t_2,\ldots,t_n\}$ (query term space)
k (# of terms to be selected)
$\Delta q \leftarrow \{\ \}$
for i = 1 **to** k **do**
$t_i^* \leftarrow argmax_{t_i \in T}\{\, r(t_i)\}$
$\Delta q \leftarrow \Delta q \cup \{t_i^*\}$
$T \leftarrow T - \{t_i^*\}$
end
Output Δq

Algorithm Reduction

Input: $T=\{t_1,t_2,\ldots,t_n\}$ (query term space)
k (# of terms to be selected)
$\Delta q \leftarrow \{\ t_1,t_2,\ldots,t_n\ \}$
for i = 1 **to** n-k **do**
$t_i^* \leftarrow argmin_{t_i \in T}\{\, r(t_i)\}$
$\Delta q \leftarrow \Delta q - \{t_i^*\}$
$T \leftarrow T - \{t_i^*\}$
end
Output Δq

Fig. 1. The Generation Algorithm and the Reduction Algorithm

2.4 Features Used for Term Selection

Linguistic and statistical features provide important clues for selection of good query terms from viewpoints of users and collections, and we use them to train function r.

Linguistic Features: Terms with certain linguistic properties are often viewed semantics-bearing and informative for search. Linguistic features of query terms are mainly inclusive of parts of speech (POS) and named entities (NE). In our experiment, the POS features comprise noun, verb, adjective, and adverb, the NE features include person names, locations, organizations, and time, and other linguistic features contain acronym, size (i.e., number of words in a term) and phrase, all of which have shown their importance in many IR applications. The values of these linguistic features are binary except the size feature. POS and NE are labeled manually for high quality of training data, and can be tagged automatically for purpose of efficiency alternatively.

Statistical Features: Statistical features of term t_i refer to the statistical information about the term in a document collection. This information could be about the term itself such as term frequency (TF) and inverse document frequency (IDF), or the relationship between the term and other terms in space T. We present two methods for estimating such term relationship. The first method depends on co-occurrences of terms t_i and t_j ($t_j \in T$, $t_i \neq t_j$) and co-occurrences of terms t_i and T-$\{t_i\}$ in the document collection. The former is called *term-term co-occur feature* while the latter is called *term-topic co-occur feature*. The second method extracts so-called context vectors as features from the search results of t_i, t_j, and T-$\{t_i\}$, respectively. The *term-term context feature* computes the similarity between the context vectors of t_i and t_j while the *term-topic context feature* computes the similarity between context vectors of t_i and T-$\{t_i\}$.

Term-term & term-topic co-occur features: The features are used to measure whether query term t_i itself could be replaced with another term t_j (or remaining terms T-$\{t_i\}$) in

T and how much the intension is. The term without substitutes is supposed to be important in *T*. Point-wise mutual information (PMI), Chi-square statistics (X^2), and log-likelihood ratio (LLR) are used to measure co-occurrences between t_i and *Z*, which is either t_j or T-$\{t_i\}$ in this paper. Suppose that *N* is the number of documents in the collection, *a* is the number of documents containing both t_i and *Z*, denoted as a = #d(t_i,Z). Similarly, we denote *b* = #d(t_i ,~Z) *c* = #d(~t_i,Z) and *d* = #d(~t_i,~Z) i.e., *Z*=*N*-*a*-*b*-*c*.

PMI is a measure of how much term t_i tells us about *Z*.

$$\mathrm{PMI}(\mathrm{t_i}, \mathrm{Z}) = \log[\mathrm{p}(\mathrm{t_i}, \mathrm{Z})/\mathrm{p}(\mathrm{t_i})\mathrm{p}(\mathrm{Z})] \approx \log[\mathrm{a} \times \mathrm{N}/(\mathrm{a}+\mathrm{b})(\mathrm{a}+\mathrm{c})] \tag{6}$$

X^2 compares the observed frequencies with frequencies expected for independence.

$$\chi^2(\mathrm{t_i}, \mathrm{Z}) = [\mathrm{N} \times (\mathrm{a} \times \mathrm{d} - \mathrm{b} \times \mathrm{c})^2]/[(\mathrm{a}+\mathrm{b})(\mathrm{a}+\mathrm{c})(\mathrm{b}+\mathrm{d})(\mathrm{c}+\mathrm{d})] \tag{7}$$

LLR is a statistical test for making a decision between two hypotheses of dependency or independency based on the value of this ratio.

$$-2\log \mathrm{LLR}(\mathrm{t_i}, \mathrm{Z}) =$$

$$\mathrm{a}\log\frac{\mathrm{a}\times\mathrm{N}}{(\mathrm{a}+\mathrm{b})(\mathrm{a}+\mathrm{c})} + \mathrm{b}\log\frac{\mathrm{b}\times\mathrm{N}}{(\mathrm{a}+\mathrm{b})(\mathrm{b}+\mathrm{d})} + \mathrm{c}\log\frac{\mathrm{c}\times\mathrm{N}}{(\mathrm{c}+\mathrm{d})(\mathrm{a}+\mathrm{c})} + \mathrm{d}\log\frac{\mathrm{d}\times\mathrm{N}}{(\mathrm{c}+\mathrm{d})(\mathrm{b}+\mathrm{d})} \tag{8}$$

We make use of average, minimum, and maximum metrics to diagnose term-term co-occur features over all possible pairs of (t_i,t_j), for any $t_j \neq t_i$:

$$f_{avg}^{X}(t_i) = \frac{1}{|T|}\sum_{\forall t_j \in T, t_i \neq t_j} X(t_i, t_j), \tag{9}$$

$$f_{max}^{X}(t_i) = \max_{\forall t_j \in T, t_i \neq t_j} \mathrm{X}(t_i, t_j) \;\; and \;\; f_{min}^{X}(t_i) = \min_{\forall t_j \in T, t_i \neq t_j} \mathrm{X}(t_i, t_j) \tag{10}$$

where *X* is *PMI*, *LLR* or X^2. Moreover, given $T=\{t_1, \ldots, t_n\}$ as a training query term space, we sort all terms t_i according to their $f_{avg}^{X}(t_i)$, $f_{max}^{X}(t_i)$, or $f_{min}^{X}(t_i)$, and their rankings varied from 1 to *n* are treated the additional features.

The *term-topic co-occur feature*s are nearly identical to the *term-term co-occur features* with an exception that *term-topic co-occur features* are used in measuring the relationship between t_i and query topic T-$\{t_i\}$. The co-occur features can be quickly computed from the indices of IR systems with caches.

Term-term & term-topic context features: The co-occurrence features are reliable for estimating the relationship between high-frequency query terms. Unfortunately, term t_i is probably not co-occurring with T-$\{t_i\}$ in the document collection at all. The context features are hence helpful for low-frequency query terms that share common contexts in search results. More specifically, we generate the context vectors from the search results of t_i and t_j (or T-$\{t_i\}$), respectively. The context vector is composed of a list of pairs <document ID, relevance score>, which can be obtained from the search results returned by IR systems. The relationship between t_i and t_j (or T-$\{t_i\}$) is captured by the cosine similarity between their context vectors. Note that to extract the

context features, we are required to retrieve documents. The retrieval performance may affect the quality of the context features and the process is time-consuming.

3 Experiments

3.1 Experiment Settings

We conduct extensive experiments on NTCIR-4 and NTCIR-5 English-English ad-hoc IR tasks. Table 1 shows the statistics of the data collections. We evaluate our methods with description queries, whose average length is 14.9 query terms. Both queries and documents are stemmed with the Porter stemmer and stop words are removed. The remaining query terms for each query topic form a query term space T. Three retrieval models, the vector space model (TFIDF), the language model (Indri) and the probabilistic model (Okapi), are constructed using Lemur Toolkit [21], for examining the robustness of our methods across different frameworks. MAP is used as evaluation metric for top 1000 documents retrieved. To ensure the quality of the training dataset, we remove the poorly-performing queries whose average precision is below 0.02. As different retrieval models have different MAP on the same queries, there are different numbers of training and test instances in different models. We up-sample the positive instances by repeating them up to the same number as the negative ones. Table 2 summarizes the settings for training instances.

Table 1. Adopted dataset after data clean. Number of each setting is shown in each row for NTCIR-4 and NTCIR-5

	NTCIR-4	NTCIR-5
	<desc>	<desc>
#(query topics)	58	47
#(distinct terms)	865	623
#(terms/query)	14.9	13.2

Table 2. Number of training instances. (x : y) shows the number of positive (x) and negative (y) MAP gain instances, respectively

	Indri	TFIDF	Okapi
Original	674(156:518)	702(222:480)	687(224:463)
Upsample	1036(518:518)	960(480:480)	926(463:463)
Train	828(414:414)	768(384:384)	740(370:370)
Test	208(104:104)	192(96:96)	186 (93:93)

3.2 Performance of Regression Function

We use 5-fold cross validation for training and testing our regression function r. To avoid inside test due to up-sampling, we ensure that all the instances in the training set are different from those of the test set. The R^2 statistics ($R^2 \in [0, 1]$) is used to evaluate the prediction accuracy of our regression function r:

$$R^2 = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{11}$$

where R^2 explains the variation between true label $y_i = (pf(T) - pf(T - \{t_i\}))/pf(T)$ and fit value $\hat{y}_i = \mathbf{w}f(t_i) + b$ for each testing query term $t_i \in T$, as explained in Section 2.2. $\bar{y}$ is the mean of the ground truth.

Table 3 shows the R^2 values of different combinations of features over different retrieval models, where two other features are taken into account for comparison. Content load (*Cl*) [14] gives unequal importance to words with different POS. Our modified content load (m-Cl) sets weight of a noun as 1 and the weights of adjectives, verbs, and participles as 0.147 for IR. Our m-SCS extends the simplified clarity score (*SCS*) [9] as a feature by calculating the relative entropy between query terms and collection language models (unigram distributions).

It can be seen that our function *r* is quite independent of retrieval models. The performance of the statistical features is better than that of the linguistic features because the statistical features reflect the statistical relationship between query terms in the document collections. Combining both outperforms each one, which reveals both features are complementary. The improvement by m-Cl and m-SCS is not clear due to their similarity to the other features. Combining all features achieves the best R^2 value 0.945 in average, which guarantees us a large portion of explainable variation in *y* and hence our regression model *r* is reliable.

Table 3. R^2 of regression model *r* with multiple combinations of training features. L: linguistic features; C1: co-occurrence features; C2: context features

Performance of Regression Model *r*		One Group of Features			Two Groups of Features			Three	Four (3+1)		All
		L	C1	C2	L&C1	L&C2	C1&C2	L&C1 &C2	m-Cl	m-SCS	
R^2	Indri	0.120	0.145	0.106	0.752	0.469	0.285	0.975	0.976	0.975	0.976
	TFIDF	0.265	0.525	0.767	0.809	0.857	0.896	0.932	0.932	0.932	0.932
	Okapi	0.217	0.499	0.715	0.780	0.791	0.910	0.925	0.926	0.925	0.926
	Avg.	0.201	0.390	0.529	0.781	0.706	0.697	0.944	0.945	0.944	0.945

3.3 Correlation between Feature and MAP

Yet another interesting aspect of this study is to find out a set of key features that play important roles in document retrieval, that is, the set of features that explain most of the variance of function *r*. This task can usually be done in ways fully-addressed in regression diagnostics and subset selection, each with varying degrees of complexity. One common method is to apply correlation analysis over the response and each predictor, and look for highly-correlated predictor-response pairs.

Three standard correlation coefficients are involved, including Pearson's product-moment correlation coefficient, Kendall's tau, and Spearman's rho. The results are given in Fig. 2, where x-coordinate denotes features and y-coordinate denotes the value of correlation coefficient. From Fig. 2, two context features, “cosine” and “cosineinc”, are found to be positively- and highly-correlated (ρ>0.5) with MAP, under Pearson's coefficient. The correlation between the term-term context feature (cosine) and MAP even climbs up to 0.8. For any query term, high context feature value indicates high deviation in the result set caused by removal of the term from the query topic. The findings suggest that the drastic changes incurred in document ranking by removal of a term can be a good predictor. The tradeoff is the high cost in feature

computation because a retrieval processing is required. The co-occurrence features such as PMI, LLR, and χ^2 also behave obviously correlated to MAP. The minimum value of LLR correlates more strongly to MAP than the maximum one does, which means that the independence between query terms is a useful feature.

In the linguistic side, we find that two features "size" and "phrase" show positive, medium-degree correlation ($0.3<\rho<0.5$) with MAP. Intuitively, a longer term might naturally be more useful as a query term than a shorter one is; this may not always be the case, but generally it is believed a shorter term is less informative due to the ambiguity it encompasses. The same rationale also applies to "phrase", because terms of noun phrases usually refer to a real-world event, such as "911 attack" and "4th of July", which might turn out to be the key of the topic.

We also notice that some features, such as "noun" and "verb", pose positive influence to MAP than others do, which shows high concordance to a common thought in NLP that nouns and verbs are more informative than other type of words. To our surprises, NE features such as "person", "geo", "org" and "time" do not show as high concordance as the others. This might be resulted from that the training data is not sufficient enough. Features "idf" and "m-SCS" whose correlation is highly notable have positive impacts. It supports that the statistical features have higher correlation values than the linguistics ones.

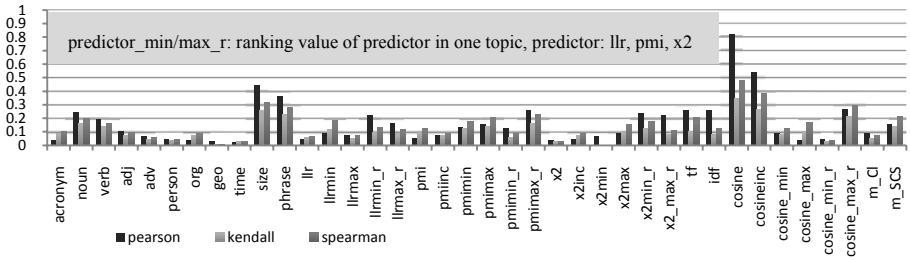

Fig. 2. Three correlation values between features and MAP on Okapi retrieval model

3.4 Evaluation on Information Retrieval

In this section, we devise experiments for testing the proposed query formulation algorithms. The benchmark collections are NTCIR-4 and NTCIR-5. The experiments can be divided into two parts: the first part is a 5-fold cross-validation on NTCIR-4 dataset, and in the second part we train the models on NTCIR-4 and test them on NTCIR-5. As both parts differ only in assignment of the training/test data, we will stick with the details for the first half (cross-validation) in the following text.

The result is given in Table 4. Evaluation results on NTCIR-4 and NTCIR-5 are presented in the upper- and lower-half of the table, respectively. We offer two baseline methods in the experiments: "BL1" puts together all the query terms into one query string, while "BL2" only consider nouns as query terms since nouns are claimed to be more informative in several previous works. Besides, the upper bound

UB is presented in the benchmark: for each topic, we permute all sub queries and discover the sub-query with the highest MAP. As term selection can also be treated as a classification problem, we use the same features of our regression function *r* to train two SVM classifiers, Gen-C and Red-C. Gen-C selects terms classified as "effective" while Red-C removes terms classified as "ineffective". Gen-R and Red-R denote our Generation and Reduction algorithms, respectively. The retrieval results are presented in terms of MAP. Gain ratios in MAP with respect to the two baseline methods are given in average results. We use two-tailed *t*-distribution in the significance test for each method (against the BL1) by viewing AP values obtained in all query session as data points, with $p<0.01$ marked ** and $p<0.05$ marked *.

Table 4. MAP of baseline and multiple proposed methods on NTCIR-4 <desc> regression model. (+x, +y) shows the improvement percentage of MAP corresponding to BL1 and BL2. TFIDF and Okapi models have PRF involved, Indri model does not. Best MAP of each retrieval model is marked **bold** for both collections.

Settings	Method	Indri	TFIDF	Okapi	Avg.
NTCIR-4	UB	0.2233	0.3052	0.3234	0.2839
<desc>	BL1	0.1742	0.2660	0.2718	0.2373
Queries	BL2	0.1773	0.2622	0.2603	0.2332
	Gen-C	0.1949**	0.2823**	**0.2946****	0.2572(+8.38%,+10.2%)
	Gen-R	0.1954**	**0.2861****	0.2875*	0.2563(+8.00%,+9.90%)
	Red-C	0.1911**	0.2755**	0.2854**	0.2506(+5.60%,+7.46%)
	Red-R	**0.1974****	0.2773**	0.2797	0.2514(+5.94%,+7.80%)
NTCIR-5	UB	0.1883	0.2245	0.2420	0.2182
<desc>	BL1	0.1523	0.1988	0.1997	0.1836
Queries	BL2	0.1543	0.2035	0.1969	0.1849
	Gen-C	0.1699**	0.2117*	0.2213*	0.2009(+9.42%,+8.65%)
	Gen-R	0.1712**	**0.2221***	**0.2232***	0.2055(+11.9%,+11.1%)
	Red-C	0.1645**	0.2194*	0.2084	0.1974(+7.51%,+6.76%)
	Red-R	**0.1749****	0.2034**	0.2160*	0.1981(+7.89%,+7.13%)

From Table 4, the MAP difference between two baseline methods is small. This might be because some nouns are still noisy for IR. The four generation and reduction methods significantly outperform the baseline methods. We improve the baseline methods by 5.60% to 11.9% in the cross-validation runs and on NTCIR-5 data. This result shows the robustness and reliability of the proposed algorithms. Furthermore, all the methods show significant improvements when applied to certain retrieval models, such as Indri and TFIDF; performance gain with Okapi model is less significant on NTCIR-5 data, especially when reduction algorithm is called for. The regression methods generally achieve better MAP than the classification methods. This is because the regression methods always select the most informative terms or drop the most ineffective terms among those that are not selected yet. The encouraging evaluation results show that, despite the additional costs on iterative processing, the

performance of the proposed algorithms is effective across different benchmark collections, and based on a query term space *T*, the algorithms are capable of suggesting better ways to form a query.

We further investigate the impact of various ranking schemes based on our proposed algorithms. The ranking scheme in the Generation algorithm (or the Reduction algorithm) refers to an internal ranking mechanism that decides which term shall be included in (or discarded away). Three types of ranking schemes are tested based on our regression function *r.* "max-order" always returns the term that is most likely to contribute relevance to a query topic, "min-order" returns the term that is most likely to bring in noise, and "random-order" returns a randomly-chosen term.

Figure 3 shows the MAP curve for each scheme by connecting the dots at (1, $MAP^{(1)}$), … , (n, $MAP^{(n)}$), where $MAP^{(i)}$ is the MAP obtained at iteration *i*. It tells that the performance curves in the generation process share an interesting tendency: the curves keep going up in first few iterations, while after the maximum (locally to each method) is reached, they begin to go down rapidly. The findings might informally establish the validity of our assumption that a longer query topic might encompass more noise terms. The same "up-and-down" pattern does not look so obvious in the reduction process; however, if we take the derivative of the curve at each iteration *i* (i.e., the performance gain/loss ratio), we might find it resembles the pattern we have discovered. We may also find that, in the generation process, different ranking schemes come with varying degrees of MAP gains. The ranking scheme "max-order" constantly provides the largest performance boost, as opposed to the other two schemes. In the reduction process, "max-order" also offers the most drastically performance drop than the other two schemes do. Generally, in the generation process, the best MAP value for each setting might take place somewhere between iteration n/2 to 2n/3*,* given *n* is the size of the query topic.

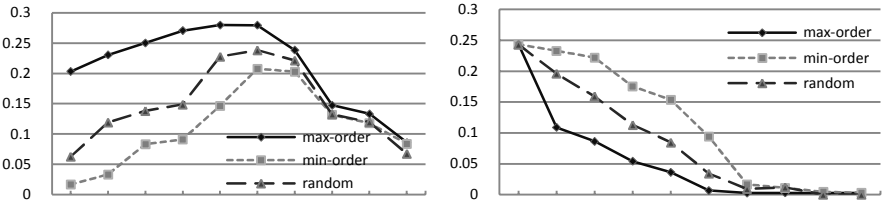

Fig. 3. MAP curves based on regression model for description queries of NTCIR-4 on TFIDF model, each with three selection order. X coordinate is # of query terms; Y coordinate is MAP.

4 Discussions and Conclusions

In this paper, we propose an approach to measure and predict the impact of query terms, based on the discovery of linguistic, co-occurrence, and contextual features, which are analyzed by their correlation with MAP. Experimental results show that our query formulation approach significantly improves retrieval performance.

The proposed method is robust and the experimental results are consistent on different retrieval models and document collections. In addition, an important aspect of

this paper is that we are able to capture certain characteristics of query terms that are highly effective for IR. Aside from intuitive ideas that informative terms are often lengthy and tagged nouns as their POS category, we have found that the statistical features are more likely to decide the effectiveness of query terms than linguistics ones do. We also observe that context features are mostly correlated to MAP and thus are most powerful for term difficulty prediction. However, such post-retrieval features require much higher cost than the pre-retrieval features, in terms of time and space.

The proposed approach actually selects local optimal query term during each iteration of generation or reduction. The reason for this greedy algorithm is that it is inappropriate to exhaustively enumerate all sub-queries for online applications such as search engines. Further, it is challenging to automatically determine the value of parameter k in our algorithms, which is selected to optimize the MAP of each query topic. Also, when applying our approach to web applications, we need web corpus to calculate the statistical features for training models.

References

1. Allan, J., Callan, J., Croft, W.B., Ballesteros, L., Broglio, J., Xu, J., Shu, H.: INQUERY at TREC-5. In: Fifth Text REtrieval Conference (TREC-5), pp. 119–132 (1997)
2. Amati, G., Carpineto, C., Romano, G.: Query difficulty, robustness, and selective application of query expansion. In: McDonald, S., Tait, J.I. (eds.) ECIR 2004. LNCS, vol. 2997, pp. 127–137. Springer, Heidelberg (2004)
3. Bendersky, M., Croft, W.B.: Discovering key concepts in verbose queries. In: 31st annual international ACM SIGIR, pp. 491–498 (2008)
4. Cao, G., Nie, J.Y., Gao, J.F., Robertson, S.: Selecting good expansion terms for pseudo-relevance feedback. In: 31st annual international ACM SIGIR, pp. 243–250 (2008)
5. Carmel, D., Yom-Tov, E., Soboroff, I.: SIGIR Workshop Report: Predicting Query Difficulty - Methods and Applications. In: Workshop Session: SIGIR, pp. 25–28 (2005)
6. Carmel, D., Yom-Tov, E., Darlow, A., Pelleg, D.: What makes a query difficult? In: 29th annual international ACM SIGIR, pp. 390–397 (2006)
7. Carmel, D., Farchi, E., Petruschka, Y., Soffer, A.: Automatic query refinement using lexical affinities with maximal information gain. In: 25th annual international ACM SIGIR, pp. 283–290 (2002)
8. Chang, C.C., Lin, C.J.: LIBSVM (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
9. He, B., Ounis, I.: Inferring query performance using pre-retrieval predictors. In: 11th International Conference of String Processing and Information Retrieval, pp. 43–54 (2004)
10. Jones, R., Fain, D.C.: Query word deletion prediction. In: 26th annual international ACM SIGIR, pp. 435–436 (2003)
11. Kumaran, G., Allan, J.: Effective and efficient user interaction for long queries. In: 31st annual international ACM SIGIR, pp. 11–18 (2008)
12. Kumaran, G., Allan, J.: Adapting information retrieval systems to user queries. In: Information Processing and Management, pp. 1838–1862 (2008)
13. Kwok, K.L.: A new method of weighting query terms for ad-hoc retrieval. In: 19th annual international ACM SIGIR, pp. 187–195 (1996)
14. Lioma, C., Ounis, I.: Examining the content load of part of speech blocks for information retrieval. In: COLING/ACL 2006 Main Conference Poster Sessions (2006)

15. Mandl, T., Womser-Hacker, C.: Linguistic and statistical analysis of the CLEF topics. In: Third Workshop of the Cross-Language Evaluation Forum CLEF (2002)
16. Mothe, J., Tanguy, L.: ACM SIGIR 2005 Workshop on Predicting Query Difficulty - Methods and Applications (2005)
17. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, Chichester (1998)
18. Yom-Tov, E., Fine, S., Carmel, D., Darlow, A., Amitay, E.: Juru at TREC 2004: Experiments with prediction of query difficulty. In: 13th Text Retrieval Conference (2004)
19. Zhou, Y., Croft, W.B.: Query performance prediction in Web search environments. In: 30th Annual International ACM SIGIR Conference, pp. 543–550 (2007)
20. Zhou, Y., Croft, W.B.: Ranking Robustness: A novel framework to predict query performance. In: 15th ACM international conference on Information and knowledge management, pp. 567–574 (2006)
21. The Lemur Toolkit:
`http://www.lemurproject.org/`