Finding and Estimating Near Optimal Queries

Wong, Wing Sze

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy

Department of Computing
The Hong Kong Polytechnic University
March, 2007

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

Wong, Wing Sze

Abstract

The ultimate objective of IR systems is to obtain optimal retrieval effectiveness. However, the best MAP values of the state-of-the-art IR systems are typically below 35% in the ad hoc automatic retrieval of TREC evaluations. This value is still far below the theoretical optimal retrieval effectiveness of 100%. In this study, we investigate whether it is possible to achieve near optimal retrieval effectiveness using the existing IR systems by formulating effective queries. These effective queries are called near optimal queries because they lead the IR systems to achieve near optimal retrieval effectiveness. Our near optimal queries are defined so as not to include the trivially good effective terms. We propose two strategies, the Idealized Relevance Feedback, and the Combinatorial Optimization Search, to find the near optimal queries under some idealized conditions. We have experimented with a substantial number of query-formulating methods based on the strategies and have evaluated these by using TREC test collections. The best MAP values of our near optimal queries for TREC-6, TREC-7 and TREC-8 test collections are 73%, 76% and 75%, respectively. It appears that a suitable choice of terms and a suitable choice of weights can substantially enhance the retrieval effectiveness of the existing IR systems. Based on the observations of the terms in the near optimal queries, we develop a classifier to estimate a near optimal query. The experimental results show that our classifier can improve the retrieval effectiveness of the user query in existing IR systems.

Acknowledgments

I would like to take this opportunity to offer my heartfelt acknowledgments to some people. This thesis could not be done without their help and support.

Firstly, I would like to especially thank my supervisor, Dr. Robert Luk, for his help and guidance. The research opportunities, ideas and advice he gave me played an important role in completing my thesis. A number of discussions with him lead me towards new and successful research directions. My deepest appreciation for his encouragement during these years of my master study.

Additionally, I would also like to thank my co-supervisor, Dr. Hong-Va Leong, for his help and suggestion. He gave me many valuable comments for improving my research studies. I appreciate his support through my master study.

Furthermore, I would like to thank my family for their never-ending care and support. This thesis is dedicated to my parents.

Lastly, I must thank my husband and my daughter. They make my life more meaningful. Their love is the most important support to me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An optimal query is a query that precisely retrieve all the relevant documents of a user's information needs (also known as a topic) from a pool of documents (also known as a collection). In other words, when an optimal query is used, an Information Retrieval (IR) system will achieve optimal retrieval effectiveness (i.e., Mean Average Precision (MAP) is 100%). Achieving optimal retrieval effectiveness is the ultimate objective of IR systems. However in practice, the best MAP values of existing IR systems are typically below 35% in the English ad hoc automatic retrieval of Text REtrieval Conference (TREC). If manual intervention is allowed in the retrieval process so as to provide relevance information to the IR system, the best MAP values are typically between 40% and 60% in the TREC evaluations [VH98, VH99, VH00]. These values are still far below 100%.

The potential causes of the poor retrieval effectiveness in existing IR systems include system factors, query factors and practical limitation factors. System factors refer to the algorithms of an IR system. Query factors refer to the selection of query terms. Practical limitation factors refer to the limitations when an IR system works in practice, such as the number of top ranked documents that a user is willing to examine. Suppose some or all of the practical limitations can be overcome in idealized situations. Then, if one or more optimal queries on a given topic can be found, it will reveal that the main cause of the poor retrieval effectiveness in an existing IR system is not the system factor but the query factor. Therefore, the principal future research question of IR systems is to examine how to find the right query terms with the right weights. Conversely, if we cannot find any optimal queries in such idealized situations, it will reveal that the main cause is the system factor. In this case, the principal future research question is to examine how to enhance existing IR models.

Two theoretical methods can be used to find the optimal queries on a given topic. Salton's optimal query [Sal71, SB88, Sal89] is a well-known method that uses Relevance Feedback (RF) to approximate the optimal query. The basic idea is to give positive weights to the terms in relevant documents and to give negative weights to the terms in non-relevant documents. Although research has discussed how Salton's optimal query can be applied in information retrieval [RHP81, Kwo87], little has been reported on how Salton's optimal query can be achieved in practice [BH03], even though more test collections from TREC and NTCIR are now available. The other straightforward method to find the optimal queries is to simply try every single possible subset of terms in the collection. However, this is very time consuming, especially for a large collection.

Our work differs from the above theoretical methods in the following ways: firstly, we report a systematic and practical study on achieving Salton's optimal query in different idealized RF situations. Secondly, we propose to investigate the near optimal queries based on the view of the combinatorial optimization. That is, the problem of finding an optimal query can be considered as the problem of searching the best combination of terms out of all the combination of terms in all the relevant documents with binary term weights. We propose to apply some local search algorithms to find the optimal queries within polynomial bounded computation times. In addition, in order to prevent trivial high retrieval effectiveness in practice, we restrict our study to examine the terms that occur in more than one document in the relevant document set (i.e., the document frequency of the selected query term should be larger than one). The optimal queries obtained in such a situation are so-called near optimal queries in our study.

The primary objective of this study is to investigate how to find and estimate near optimal queries. More specifically, we will:

- determine the main cause of the poor retrieval effectiveness in existing IR systems;
- apply Salton's optimal query as well as Relevance Feedback to find the near optimal queries;
- find the near optimal queries based on the view of the combinatorial optimization search;
- investigate what and how the near optimal query terms and associated weights would lead to the near optimal retrieval effectiveness;
- develop a mechanism to estimate the near optimal queries of a user's information needs.

The main contributions of this dissertation are as follows:

- Most of the near optimal queries about the topics in our test collections have been found under some idealized situations. It appears that the main cause of the poor retrieval effectiveness in existing IR systems is the query factor. We believe that a suitable choice of terms and a suitable choice of weights can substantially enhance the retrieval effectiveness of existing IR systems.
- A systematic and practical study of the impact of different idealized Relevance Feedback situations (which is called IRF in our study) on finding the near optimal queries is reported. Our IRF performs statistically significantly better than the corresponding practical RF which is constrained by the set of practical limitations.
- Many term ranking functions are examined and some novel term ranking functions are proposed in order to provide a better ranked list for term selection.

- A Combined Search method is proposed that can be used to find the near optimal queries faster than our simulated annealing method. This method achieves better retrieval effectiveness than our IRF.

- Many characteristics of the near optimal query terms are examined and discussed.

- A Good Query Term Extractor (GQTE) is proposed that can be used to estimate the near optimal query regarding a user's information need.

- A novel method to define the query size by the GQTE's confidence value is introduced. The query size will be different for different topics.

A review and comparision of four existing IR models is given in Chapter 2 that illustrates that the main cause of the low retrieval effectiveness in existing IR systems may not be the system factor. Moreover, Chapters 3 and 4 examine the effects on the query factors and the practical limitation factor by finding the near optimal queries in idealized situations. In Chapter 3, our IRF is proposed to approximate the near optimal queries in idealized situations. Five idealized assumptions are explored, and several issues related to RF are examined, such as what terms to select and how many terms to select in each relevance feedback cycle. In Chapter 4, a novel Combined Search method is introduced to find the near optimal queries based on the concepts of Combinatorial Optimization Search. The performance of our Combined Search method is compared with several well known Combinatorial Optimization Search methods, such as Hill Climbing, Best First Search and Simulated Annealing. The experimental results in Chapters 3 and 4 tell us that the main cause of the low retrieval effectiveness in existing IR systems is the query factor. Additionally, in order to investigate how to find the effective query terms with the effective weights, Chapter 5 explores the characteristics of those near optimal query terms that there achieved in Chapters 3 and 4. Based on these observations, Chapter 5 further introduces a novel Good Query Term Extractor that can be used to estimate the ‘Good’ query terms, and to generate the estimated near optimal query about a user’s information needs. Several term weight schemes and confidence levels are examined to decide the suitable query weights and query size. Finally, Chapter 6 concludes with suggestions some possible further developments.

Chapter 2

Effects on system factors

This chapter focuses on examining the effects on the system factors, which is one of the potential causes of the poor retrieval effectiveness in existing IR systems. This chapter begins by reviewing various state-of-the-art retrieval models in IR literature, as well as several evaluation measures for retrieval effectiveness. Then in Section 2.2, we describe the experimental set up, including our test collections, evaluation measures and our IR system with four state-of-the-art retrieval models. This environment is used for the subsequent experiments in this study. Section 2.3 illustrates that our retrieval models perform as well as those in TREC. Section 2.4 examines the effects on the system factors by investigating the correlation between query performance and retrieval models. The experimental results tell us that the main cause of the poor retrieval effectiveness in existing IR systems may not be the system factors. Furthermore, the experiments in Section 2.5 help us to establish our baseline retrieval model and our baseline retrieval effectiveness for the subsequent experiments in this study. Finally, a summary of this chapter is given in Section 2.6.

2.1 Literature Review

It this section, two kinds of literature reviews are given. Firstly, several retrieval models in theoretical IR are reviewed and the corresponding state-of-the-art retrieval systems are introduced. Then, some evaluation measures of retrieval effectiveness are discussed.

2.1.1 Retrieval Models

Three major theoretical retrieval models in IR are Boolean Model, Vector Space Model and Probabilistic Model. The Boolean Model is based on set theory and Boolean algebra, in which the documents are sets of terms and queries are Boolean expressions of terms. Roughly speaking, the Boolean Model only retrieves the exactly matched documents. Therefore, few participants of TREC use the Boolean Model. In the Vector Space Model (VSM), both documents and queries are vectors over a set of term weights. The similarity of a document vector to a query vector is equal to the cosine of the angle between them. There are many systems using VSM that have achieved outstanding performance in TREC, such as the SMART system [Sin97]. The Probabilistic Model uses probability theory to model the uncertainty in the retrieval process, in which the documents are ranked in decreasing order of probability of relevance to the query. Logistic Regression Model (LR), Pircs Retrieval Model (PIRCS) and 2-Poisson Model are three famous examples of the Probability Model. LR [GC97] estimates the weights of the Bayesian probabilistic model in a principled manner by using regression to model the dependencies among the data, whereas PIRCS [KGX97] calculates the document weight as a linear weighted sum of the activation of query terms and documents in a conceptual network of query terms, index terms and documents. The 2-Poisson Model uses the 2-Poisson distribution to model the dependencies among the data. The 2-Poisson Model with BM25 [WRB$^+$97] is a model that has often shown to perform well at TREC, and to perform better than other retrieval models.

2.1.2 Evaluation Measures

Precision and recall are two common evaluation measures for retrieval effectiveness in IR. Precision is a measure of the ability of an IR system to present only relevant documents. It is computed by the number of relevant documents retrieved and then divided by the total number of documents retrieved, whereas, the recall is a measure to present all relevant documents. It is calculated by the number of relevant documents retrieved and then divided by the number of relevant documents in the collection.

Mean Average Precision (MAP) and R-precision (R-prec) are two popular evaluation measures used in TREC. MAP is the average of the precision values obtained after each relevant document is retrieved. It is helpful to compare the performance of different systems when the retrieved results are a ranked list of documents. For example, consider a query that has three relevant documents which are retrieved at ranks 1, 2 and 4. The precision obtained when each relevant document is retrieved is 1, 1 and 0.75 respectively, the mean of which is 0.92. Therefore, MAP for this query is 0.92.

R-prec is the precision after R documents have been retrieved, where R is the number of relevant documents for the topic. It de-emphasizes the exact ranking of the retrieved relevant documents, which can be useful when there are large numbers of relevant documents in the collection. For example, assume the number of relevant documents for a topic is three. If the query returns 2 relevant documents in the top 3 documents for the topic, then the R-prec for this query is 0.67.

2.2 Experimental Setup

In this section, the test collections and the evaluation measures in our experiments are introduced. Our indexing method of the terms in collection is mentioned. Four types of state-of-the-art retrieval models in IR are discussed and implemented for subsequent experiments.

2.2.1 Test Collections and Evaluation Measures

The test collection in TREC is the most commonly used collection in IR research. It can be used to evaluate the performance of an IR system by using the test topics and the corresponding offical retrieval results. In general, there are two main tasks in TREC: the ad hoc task, and the routing task. The ad hoc task investigates how the IR system uses standing queries to search for new documents, whereas the routing task investigates how the IR system searches for a static set of documents using new topics. In our study, we use the ad hoc task rather than the routing task because we focuses on examining how to formulate the effective queries rather than on how to cluster the documents. In order to have more observation points and to investigate the effects on different collections, three of the TREC collections in English ad hoc tasks are selected for our experiments: TREC-6, TREC-7 and TREC-8.

The TREC collection in ad hoc task includes three parts: documents, topics and relevance judgments. There are 556,077 documents in TREC-6 (nearly 2139 megabytes), and a subset of 528,155 documents in both TREC-7 and TREC-8 (nearly 1904 megabytes). The TREC-6 test collection contains Congressional Records documents that are not found in TREC-7 and TREC-8. Additionally, each test collection contains 50 topics. The topic represents the information need of the user. This includes three sections: Title field (T), Description field (D), and Narrative field (N). The common methods to formulate the user query are: long query, which uses all the fields in the topic (i.e., TDN query); and title query, which uses the title field only (i.e., T query). Furthermore, the relevance judgments include a set of official relevant documents for each topic. Detailed information is shown in Table 2.1. It should be noted that the stop words are excluded in any type of measures regarding the number of terms.

Two evaluation measures, MAP and R-precision, are used for our experiments.

	TREC-6	TREC-7	TREC-8
Collections			
Total number of documents	556077	528155	528155
Total number of unique terms	2049150	1791942	1797942
Relevant Documents			
Avg number of documents	92	93	95
Avg number of unique terms	8725	6523	6987
Queries			
Avg number of terms in T	2	2	2
Avg number of terms in TDN	57	37	37

Table 2.1: A summary of our test collections

In addition, our experiments index the documents using strings between two space characters as index terms. Unwanted words are filtered using a list of 441 stop words, and candidate index terms are stemmed by the Porter stemming algorithm [Por80].

2.2.2 Implementation of Retrieval Models

We have implemented four types of state-of-the-art retrieval models in IR: VSM with pivoted unique normalization, LR, PIRCS and 2-Poisson Models with BM25, to examine whether different retrieval models will perform substantial differently or similarly, and to demonstrate that the poor retrieval effectiveness of these state-of-the-art retrieval models is not caused by our implementation skills. In addition, Pseudo Relevance Feedback (PRF), collection enrichment and merge list methodologies are not used in our experiments because we want to examine direct retrieval effectiveness. Otherwise, techniques of selecting terms in the top ranked documents or from other collections may have an unknown impact on the evaluations.

Vector Space Model

Our VSM uses the pivoted unique normalization [SBM96] to compute the similarity score $sim(d, q)$ (also known as document weighting function) between the query q

and the document d as follows:

$$sim(d,q) = \sum_{t\in q} \omega_(d,t) + \sum_{t\in q} \omega(q,t) \tag{2.1}$$

where,

$$\begin{aligned}
\omega(d,t) &= Lnu = \frac{1+\log(TF_(d,t))}{1+log\frac{\sum\limits_{d\in C} TF(d,t)}{\|C\|_1}} \times \frac{1}{1-\delta+\delta\times\frac{\|d\|_1}{\triangle}} \\
\omega(q,t) &= TF(q,t)
\end{aligned}$$

where $\omega(d,t)$ is the document weighting function and $\omega(q,t)$ is the query weighting function. $TF(d,t)$ and $TF(q,t)$ are the term frequency of the term t in the document d and in the query q, respectively. δ is the slope and is set to 0.2 in our experiments. $\|d\|_1$ is the City-Block length of the document d (i.e., the number of terms in the document d), and $\|C\|_1$ is the City-Block length of the collection C. $\triangle$ is the pivot which is the mean number of terms in a document (i.e., $\triangle = \frac{\sum_{d\in C}\|d\|_1}{card(C)}$), $card(C)$ is the number of documents in the collection C. This similarity calculation is the same as the one used by AT & T in TREC [Sin97] except the query weighting function. We use the query term frequency as the query weights rather than using ltu formula (i.e., $ltu = \sum_{t\in q}(1+\log(TF_(q,t))) \times \log(\frac{card(C)+1}{DF(C,t)}) \times \frac{1}{1-\delta+\delta\times\frac{\|q\|_1}{\triangle}}$, where $DF(C,t)$ is the number of documents in the collection C containing term t).

Logistic Regression Model

The similarity function of our LR is similar to UC Berkeley's at TREC [GC97]. The formula is shown in Equation 2.3 where $\|q\|_1$ is the City-Block length of the query q:

$$sim(d,q) = \frac{1}{1+e^{-\log O(R|d,q)}} \tag{2.2}$$

where,

$$\begin{aligned}
\log O(R|d,q) &= -3.51 + \frac{1}{\sqrt{card(C)}+1}\Phi + 0.0929 \times card(C) \\
\Phi &= 37.4 \times \sum_{t\in q} \frac{TF(q,t)}{\|q\|_1 + 35} \\
&+ 0.33 \times \sum_{t\in q} \log \frac{TF(d,t)}{\|d\|_1 + 80} \\
&- 0.1937 \times \sum_{t\in q} log \frac{\sum_{d\in C} TF(d,t)}{\|C\|_1}
\end{aligned}$$

Pircs Retrieval Model

Our PIRCS is similar to PIRCS at TREC [KGX97]. In order to ignore the effect on the query terms, we simply set $\alpha = 1$:

$$sim(d,q) = \alpha \sum_{t\in q} \left(\frac{1}{1+e^{-TF(q,t)}} \times \omega_(d,t) \right) + (1-\alpha) \sum_{t\in q} \left(\frac{1}{1+e^{-TF(d,t)}} \times \omega(q,t) \right) \tag{2.3}$$

where,

$$\begin{aligned}
\omega(d,t) &= \log \left(\frac{TF(d,t)}{\|d\|_1 - TF(d,t)} \times \frac{\|C\|_1 - \|d\|_1 - \sum_{d\in C} TF(d,t) + TF(d,t)}{\sum_{d\in C} TF(d,t) - TF(d,t)} \right) \\
\omega(q,t) &= \log \left(\frac{TF(q,t)}{\|q\|_1 - TF(q,t)} \times \frac{\|C\|_1 - \sum_{d\in C} TF(d,t)}{\sum_{d\in C} TF(d,t)} \right)
\end{aligned}$$

2-Poisson Model

There are many variations of 2-Poisson Model, such as using BM11 or BM25 weighting function. We use the Okapi BM25 weighting function [WRB$^+$97] to compute the document weight and use Euclidean distance to calculate the document length. Moreover, we use the similarity function in ACSys [HTC97] rather than in OKAPI because ACSys has achieved the best results in TREC-6. We set $\alpha = 1$ and $\beta = 1$ in our experiments and named this model BM25:

$$sim(d, q) = \sum_{t \in q} \frac{\omega(d, t) \times TF(d, t)}{\alpha \times \left(1 - \beta + \beta \times \frac{\|d\|_{eu}}{\|d\|_{eu}/\|d\|_1}\right) + TF(d, t)} \tag{2.4}$$

where,

$$\omega(d, t) = \log\left(\frac{card(C) - DF(t) + 0.5}{DF(t) + 0.5}\right)$$

2.3 Evaluation of our retrieval models

This section illustrates that our retrieval models are similar to the current state-of-the-art retrieval models in TREC. This experiment uses TDN queries to investigate the retrieval effectiveness of our four retrieval models: VSM, LR, PIRCS and BM25. The reason for choosing TDN queries is because these are usually the better retrieval effectiveness queries than their corresponding title queries.

The comparison of the retrieval effectiveness of our retrieval models and those known in TREC is shown in Table 2.2. This comparison is based on using the same techniques in automatic ad hoc retrieval, such as no PRF and no document enrichment methods were applied. The dashes shown in Table 2.2 refers to no related results in TREC. From Table 2.2 we can see that the MAP values of our retrieval models are near to those known in TREC. Therefore, it is believed that our implementations of the retrieval models are similar to those in TREC. Moreover, our VSM achieves 0.231, 0.236 and 0.273 MAP value in TREC-6, TREC-7 and TREC-8, respectively. This illustrates that the performance of our VSM is both stable and is comparable to the existing system.

2.4 Effects on Retrieval Models

This section examines whether different retrieval models will perform substantially differently or similarly, as well as illustrates that the main cause of the poor retrieval

	TREC-6		TREC-7		TREC-8	
Models	Known	Ours	Known	Ours	Known	Ours
VSM	0.218	0.231	0.218	0.236	0.259	0.273
LR	0.205	0.198	-	0.199	-	0.255
PIRCS	-	0.229	0.214	0.219	-	0.256
BM25	0.246	0.241	0.248	0.236	-	0.284

Table 2.2: Comparing the MAP values of our four retrieval models and those known in TREC by using TDN queries

effectiveness in existing IR systems may not be the system factors. If the retrieval effectivenesses of the same queries in different retrieval models are correlated, this may imply that the poor/good retrieval results are not necessarily caused by the models, but can be caused by the queries.

This experiment uses a novel evaluation measure the relative score $RS(q)$, to examine the correlation query performance among the retrieval models. The equation of the relative score is shown in Equation 2.5, in which $\|M\|_1$ is the total number of the test retrieval models M, whereas, $RC(q)$ is the number of the retrieval models that achieve better or equal MAP values than a threshold value using the query q. This threshold value is defined as the average MAP value of 50 topics in the collection by using one of the test retrieval models. If the relative score is equal to 1 or -1, it means that all of the retrieval models achieve good retrieval effectiveness or poor retrieval effectiveness, respectively. Therefore, if there are many queries that achieve 1 or -1 value, it reveals that the test retrieval models perform similarly:

$$RS(q) = \frac{2RC(q) - \|M\|_1}{\|M\|_1} \tag{2.5}$$

In this experiment, four retrieval models: VSM, LR, PIRCS and VSM; three test collections: TREC-6, 7 and 8; and three types of query: T, TD and TDN (totally 150 queries for each query type) are used, where TD and TDN are long queries that are produced by concantenating the title with the description field and with the description together with the narrative field of the topic, respectively. The threshold

	TREC-6	TREC-7	TREC-8
T	0.211	0.185	0.247
TD	0.213	0.203	0.272
TDN	0.231	0.236	0.273

Table 2.3: The thresholds for the three test collections and the three types of queries

values used in this experiment are shown in Table 2.3. These values are the average MAP values of 50 topics in each collection by using the VSM model. Since the current number of retrieval models $\|M\|_1$ is four, if the query q achieves better or the same threshold value in 4, 3, 2, 1, 0 retrieval model(s), the relative score $RS(q)$ will equal to 1, 0.5, 0, -0.5, -1, respectively.

The query ratio of 3 types of queries against different relative score values $RS(q)$ is shown in Figure 2.1. This query ratio is defined as the number of queries divided by 150. From the figure we can see that most of the queries achieves $RS(q) = -1$ (i.e., above 50%), and $RS(q) = 1$ (i.e., above 30%) for T, TD or TDN queries. In other words, a good majority (i.e., 80%) of the queries that achieved good or poor MAP values are independent of the use of different retrieval models. Thus, we believe that the main cause of the poor retrieval effectiveness in existing IR systems may not be the system factors.

2.5 Establishing the baseline

Section 2.4 illustrated that the good or poor MAP values are independent of the use of our retrieval models. This implies that we can choose one of our retrieval models as the baseline model for the subsequent experiments. According to the experiment results in Section 2.3, our VSM achieves similarly good retrieval effectiveness compared with other test retrieval models and state-of-the-art VSM systems in TREC. Therefore, we choose our VSM as our baseline retrieval model for the subsequent experiments in this study.

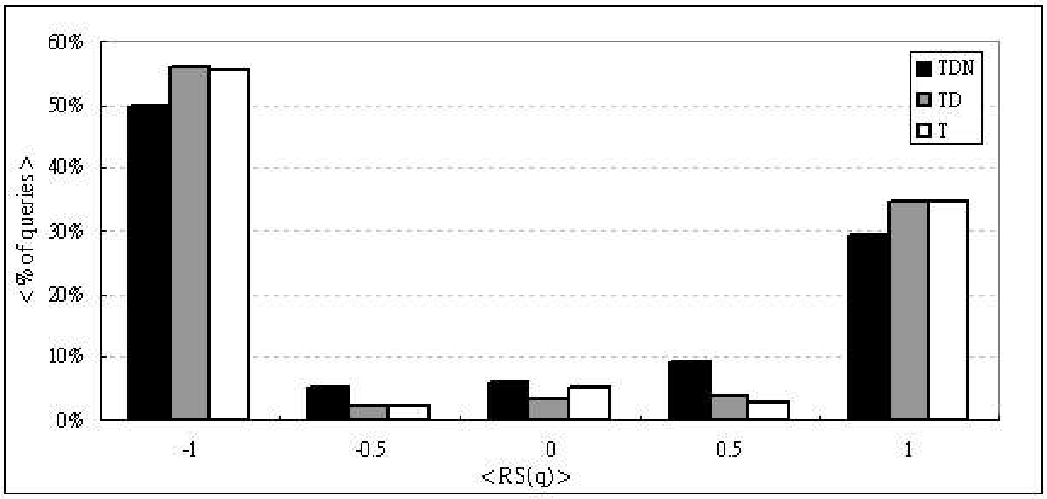


Figure 2.1: The query ratio of 3 types of queries against different relative scores

2.6 Summary

In this chapter, we have described our experimental setup as well as our four state-of-the-art retrieval models. We have illustrated that our retrieval models are performing as good as those in TREC. Also, the results on examining the correlation between query retrieval effectiveness and different retrieval models tell us that the queries achieved good or poor MAP values independent of the use of different retrieval models. This reveals that the main cause of the poor retrieval effectiveness in existing IR systems may not be the system factors. Therefore, we will mainly focus on examining the query factors and practical limitation factors in the following chapters. Finally, we have decided to use our VSM as the baseline retrieval model for the subsequent experiments in this study.

Chapter 3

Idealized Relevance Feedback Strategy

After examining the effects on the system factors in Chapter 2, this chapter focuses on examining the effects on the query factors and the practical limitation factors, which are the other two potential causes of the poor retrieval effectiveness in existing IR systems. We believe that if some or all of the practical limitations can be overcome in idealized situations and if one or some optimal queries on a given topic can be found, it will reveal that the main cause of the poor retrieval effectiveness in existing IR systems is the query factor. Therefore, this chapter is devoted to the study of finding the optimal queries in idealized situations.

This chapter begins with a review of various definitions of optimal query, as well as the approximation methods in the IR literature. Then a detailed review on Relevance Feedback (RF) is given. RF is one of the most well-known methods to approximate Optimal Query and is used in this chapter. Afterwards, Section 3.2 describes the definitions of our approximation method, called Idealized Relevance Feedback (IRF). Five related assumptions for relaxing the practical limitations are given and are examined in turn. First, Section 3.3 examines the impact of the user bias assumption (1) by comparing the effects of having and not having user query terms on retrieval effectiveness. Second, Section 3.4 examines the dimensionality reduction assumption (2) by comparing the effects of different term selection functions and different query sizes on retrieval effectiveness. Third, Section 3.5 examines the asymptotic assumption (3) by comparing the retrieval effectiveness of IRF and practical RF. Then, Section 3.6 examines the unweighted assumption (4) by investigating the effects of positively weighted terms on retrieval effectiveness. Next, Section 3.7 examines the non-negative weight assumption (5) by exploring the effects of negatively weighted terms on retrieval effectiveness. Finally, Section 3.8 summarizes the findings in this chapter.

3.1 Literature Review

This section begins by introducing three different definitions of the optimal query in IR literature. This is followed by reviewing Relevance Feedback, which is one of the methods used to approximate the optimal query.

3.1.1 Optimal Query

The optimal query can be defined differently according to different retrieval models in IR, such as the Boolean Model, Vector Space Model and Probabilistic Model. From the view of the Boolean Model, the optimal query can be defined as a set of Boolean expressions of terms which will exactly match all the known relevant documents, whereas, from the view of the Probabilistic Model, the optimal query can be defined as ”one which will recover all the known relevant documents of a query in their best probability of relevance ranking” [Kwo87]. The most popular definition of optimal query is in the view of the Vector Space Model and was first proposed in the SMART system by Salton [Sal71], in which the optimal query is defined as a vector which maximizes the difference in query-document correlation between relevant and non-relevant document subsets. If we use cosine correlation between two vectors as the distance function to calculate query-document correlation, then the optimal query can be defined as in Equation 3.1, which is also called

Salton's Optimal Query q_s.

$$q_s = \frac{1}{card(R)} \sum_{d \in R} \frac{d}{\|d\|_1} - \frac{1}{card(S)} \sum_{d \in S} \frac{d}{\|d\|_1} \tag{3.1}$$

where d is the document vector, R and S are the set of all the relevant documents (vectors) and all the non-relevant documents (vectors) for a particular topic, respectively. $card(.)$ returns the number of documents in the set. The researchers [Sal71, Roc71] proposed that using Relevance Feedback with respect to retrieval output can approximate this optimal query.

3.1.2 Relevance Feedback

Relevance Feedback (RF) is a popular and effective query reformulation technique for improving retrieval effectiveness since its initial conception by Rocchio [Roc71, Rob90] in the 1960's. RF is used as a mechanism to estimate the practical best retrieval effectiveness (called performance limits of retrieval in our study) of automatic ad hoc retrieval [Wil96, VH98] because RF has been demonstrated to enhance retrieval effectiveness in many studies.

RF modifies the query iteratively, based on the user's judgments of the top retrieved documents. This approach is based on two hypotheses: firstly, some of the retrieved documents are relevant documents; and secondly, the term-weight vectors of the documents identified as relevant to a given query have similarities among themselves. Furthermore, it is assumed that non-relevant documents have term-weight vectors that are not similar to the ones for the relevant documents. Therefore, reformulating the query with relevance information can shift the query vector toward the term-weight vector space of the relevant documents. According to Salton [Sal71] and some variations proposed by Rocchio [Roc71], the query vector q_m at the m-th RF iteration is defined as in Equation 3.2, in which q_{m-1} is the previous query vector which is generated at the $(m-1)$-th RF iteration, U_m and V_m is the set of relevant documents (vectors) and the set of non-relevant documents (vectors) identified by the user up to the m-th iteration for a particular topic, respectively, and α, β and γ are parameters. Rocchio [Roc71] proposed to set $\alpha = 1$ and $\beta > \gamma$. This is because firstly, he believed that the previous query vector (i.e., original query) contained important information that should not be removed. Secondly, he believed that the information in the relevant documents is more important than the information in the non-relevant documents. A variant which is proposed by Ide [Ide71] sets equal weights to the previous query vector, retrieved and relevant document vectors and retrieved and non-relevant document vectors (i.e., $\alpha = \beta = \gamma = 1$) and ignores $card(U_m)$ and $card(V_m)$, According to the experimental results from Salton and Buckley [SB90], Ide's method achieves better retrieval effectiveness than others:

$$q_m \equiv \alpha q_{m-1} + \frac{\beta}{card(U_m)} \sum_{d \in U_m} d - \frac{\gamma}{card(V_m)} \sum_{d \in V_m} d \tag{3.2}$$

It is well known that it takes time and effort to obtain relevance judgment from users. Therefore, another more efficient and automatic approach is called Pseudo Relevance Feedback (PRF), which assumes that the top ranked documents retrieved are relevant, so that user judgments are not needed. PRF is the most popular query expansion method in blind feedback or automatic ad hoc retrieval. However, it is difficult to decide the number of top ranked documents [Har92].

With respect to term selection, it is obvious that Equation 3.2 uses all the terms in retrieved relevant documents U_m and retrieved non-relevant documents V_m to formulate the query q_{m+1} is very time consuming. Harman [Har92] have illustrated that the queries expanded with 20 terms can achieve better retrieval effectiveness than the queries expanded with all the terms in the retrieved relevant documents. This means that we need a good term selection method to select the good effective terms from the relevant and non-relevant documents. In fact, there are many term selection methods proposed in the IR literature [YLC76, All96, RHP81, SB88, SB90, Eft93, Dun97, CR02, FWX04, CL04, KK04, GL04]. We simply classify them into three classes:

- Manual Selection: in this method, those selected terms are decided by the users. It seems that this is a good term selection approach because users know what they want. However, Magennis and van Rijsbergen [MR97] have shown that Manual Selection is not a particularly good approach because the user cannot understand the exact term distribution in the collection. Even for the experienced user, he also cannot always produce the best selection. Therefore, Manual Selection may not be able to obtain good retrieval performance consistently for all queries.

- Top-N: this is an automatic and effective approach to select the terms based on top ranked significant terms. This significance of a term can be calculated in many manners. The most popular method is TF-IDF [SB88], where TF [Luh58] denotes the term frequency property that is local and content-oriented to relevant documents, while IDF [Jon72] denotes the inverse document frequency that is global and discrimination-oriented for the collection. However, this method is sensitive to the size of relevant documents [MO01]. Several enhancements are proposed, such as [Kwo96] which normalizes the size of relevant documents to reduce the sensitivity, and $W4$ (i.e., $F4$) [RJ76] which normalizes the size of both relevant documents and collection. Another method applies the information gain concept to select significant terms [CMRB01, HMIH99]. In our study, we explore and propose many term selection methods for the Top-N approach because we believe that this approach is a stable, efficient and effective approach.

- Clustering: Top-N approach will carry a risk when selecting the terms which are not directly related to the user information need but appear frequently in relevant documents. This will happen if the relevant documents have multiple topics but only one of the topics is related to the user information need. In this case, Clustering is a useful approach to solve this problem. The basic idea of Clustering [AF77] is to cluster the relevant documents into several topics, and then select all the terms from the cluster, which is most related to the user information need, to formulate the new query. Instead of using one topic in the relevant documents, [XC96] have proposed to use the noun group which is in the top ranked passage of the relevant documents, and [LAJ01] have proposed to use the summary of the relevant documents, both of them have achieved improved results. However, we do not consider Clustering approach in our study because we want to examine the retrieval effectiveness of the IR model without any pre-processing techniques.

3.2 Definition of Idealized Relevance Feedback

We propose to use RF in idealized situations to find the optimal query. Here, 'idealized' means that some of the relevance feedback constraints have been relaxed or some of the practical limitations have been overcome. For example, we assume that all the relevant documents are discovered as the number of iterations of RF tends to infinity. This assumption is likely to be valid by two approaches: (1) if the user examines the entire retrieval list rather than just the top ten or twenty documents in each RF iteration. Such RF in idealized situations is called Idealized RF (IRF) in our study. Or (2) if Pseudo RF (PRF) has a classifier that identifies all relevant documents from the entire retrieval list without errors. Such PRF in idealized situations is called Idealized PRF (IPRF).

We are interested in examining the retrieval effectiveness of IRF as well as IPRF in this chapter for several reasons:

1. If IRF or IPRF can achieve near optimal retrieval effectiveness, it means that the main cause of the poor retrieval effectiveness in existing IR systems is the query factor. Therefore, the principal future research question of IR systems is to examine how to find the effective query terms with appropriate weights.

2. If IRF performs substantially better than the corresponding practical RF, it

may be worthwhile investigating how to reduce the practical limitations of RF. However, there is neither a theoretical nor a practical quarantee that IRF will perform better than RF. We formulate our IRF hypothesis for verification as follows:

> *IRF Hypothesis (1): The retrieval effectiveness of IRF is at least as effective as its practical RF version.*

3. If the IRF Hypothesis (1) is confirmed, this naturally leads to the IPRF hypothesis stated without proof:

   > *IPRF Hypothesis (2): The retrieval effectiveness of IPRF is at least as high as that of its practical PRF version.*

4. If the IRF Hypothesis (1) is confirmed, it would be less labor intensive and better to estimate retrieval effectiveness limits using IRF rather than using RF. In some TREC workshops, RF is used to estimate the performance limits of ad hoc retrieval (e.g., [VH98]). However, due to user idiosyncrasies and the information gap between the user's knowledge and the information in the documents, the user may not be making the identical relevance judgment as the TREC evaluators. The amount of user effort is also limited. Therefore, there is some potential that IRF provides an alternative method to estimate the performance limits of automatic ad hoc retrieval because the relevant judgement is hard in the TREC evaluations.

5. If IRF performs substantially better than practical RF, it may be possible to build test collections by using an approximation of IRF. It would allow the formulation of a better query and the discovery of more relevant documents for the user to identify as the last iteration.

6. By examining IRF and IPRF, we can separate the problems of (P)RF better and therefore we can identify the problems of (P)RF better in this way.

For example, we can identify the better term selection method by asserting assumption (2) of IRF in Table 3.1. In this case, problems due to limited user effort and the user variability, can be isolated and studied by designing an experiment where practical limitations due to other assumptions (e.g., assumption (3) in Table 3.1) are overcome.

7. The current sizeable test collections, such as TREC and NTCIR, provide an opportunity to obtain more observations about (P)RF and query formulation apart from using earlier collections like Cranfield, CACM, etc., as well as the earlier techniques to find better queries using genetic algorithms [LPGBA02, LPGBA03].

The rest of this section is organized as follows. Firstly the definition of our IRF is given. Five assumptions about the practical limitations are introduced in turn. Moreover, the evaluation measure of the performance limit of IRF (i.e., the effectiveness of the optimal query) is mentioned. Finally, the term selection method for formulating optimal query, named greedy term selection method, is presented.

3.2.1 Definition and Assumptions

Our IRF is based on Salton's optimal query q_s [Sal71, Sal89] (Equation 3.1), which transcends the practical limitations of RF summarized by the set of assumptions in Table 3.1. First, we assume that the optimal query q_o for a user is displaced slightly by his/her preferences expressed by his/her query vector q (i.e., $q_o \equiv q_s + q$). This displacement accounts for the idiosyncrasies of the user's graded relevance judgment but it is assumed that this displacement only affects the ranking of the documents but not the ultimate retrieval effectiveness performance. The RF biased by the user query q can be expressed as the following successive approximation to

q_o as:

$$\begin{cases} f_m \equiv 0 & \text{for } m = 0 \\ f_m \equiv \alpha f_{m-1} + \dfrac{\beta}{card(U_m)} \displaystyle\sum_{d \in U_m} d - \dfrac{\gamma}{card(V_m)} \displaystyle\sum_{d \in V_m} d & \text{for } m > 0 \end{cases} \tag{3.3}$$

$$q_m \equiv q + (1 - \alpha) f_m \quad \text{for } m \geq 0 \tag{3.4}$$

where q_m is the query vector generated at the m-th iteration, f_m is the feedback vector at the m-th iteration, and d is the document vector. U_m is the set of relevant documents (vectors) identified by the users up to the m-th iteration, V_m is the set of non-relevant documents (vectors) identified by the user up to the m-th iteration, $card(.)$ returns the cardinality of the set, and α, β and γ are parameters.

Second, it is assumed that document d is represented by another vector d' which has a subset of terms in d. The difference in retrieval effectiveness using d and d' is assumed not to be substantial as suggested by some researches (such as [Har92]) in the literature.

Third, as m tends to infinity, it is assumed that $U_\infty = R$ and $V_\infty = S$, where R and S are the sets of all the relevant documents and non-relevant documents for a particular topic, respectively. As a result, the asymptotic query q_∞ is equal to q_o:

$$q_o \equiv \alpha q + \frac{\beta}{card(R)} \sum_{d' \in R} d' - \frac{\gamma}{card(S)} \sum_{d' \in S} d' = q_\infty \tag{3.5}$$

Fourth, it is assumed that terms in the document vectors have unity weights. Equation 3.5 is further simplified to the following using set notations instead:

$$q_\infty = \alpha q \cup \frac{\beta}{card(R)} \bigcup_{d' \in R} d' - \frac{\gamma}{card(S)} \bigcup_{d' \in S} d' \tag{3.6}$$

It might be obvious that Salton's optimal query q_s will always achieve good (or near optimal) results because there are many negatively weighted terms in the non-relevant documents but not in the relevant documents, so that the non-relevant documents are filtered by these negatively weighted terms. However, if these negatively weighted terms are discarded, it becomes interesting to know the performance

#	Assumption Name	Description or condition
1	User Bias	$q_o \equiv q_s + q$ assuming that the retrieval effectiveness of q_o and q_s are equally optimal while the document ranking of q_o is preferred by the user.
2	Dimensionality Reduction	d is represented by another vector d' which has a subset of terms in d and the difference in retrieval effectiveness using d and d' is not substantial.
3	Asymptotic	$m \to \infty \Rightarrow U_m = R \wedge V_m = S$
4	Unweighted	Terms in the document vectors have unity weight.
5	Non-negative Weight	$\gamma = 0$

Table 3.1: IRF Assumptions

limits with just positively weighted or unweighted terms, as used by some PRF algorithms (e.g., [BAS93]). Therefore, the fifth assumption is assumed that negatively weighted terms are discarded (i.e., $\gamma = 0$). If $\alpha = \beta = 1$, the asymptotic query q_∞ is simplified to the Equation 3.7. This is our basic IRF equation:

$$q_\infty = q \cup \bigcup_{d' \in R} d' \tag{3.7}$$

3.2.2 Greedy Term Selection algorithm

According to assumption (2) in Table 3.1, q_∞ can be approximated by a smaller query $q_{\infty,n}$ of n terms that can obtain performance similar to $card(q_\infty) \geq n$. The term selection algorithm should find the best performing query $q_{\infty,\max}$ as an approximation of q_∞. The performance limit MAP_o of our basic IRF (Equation 3.7) is defined as:

$$\begin{aligned} MAP_o &\equiv MAP(q_{\infty,\max}) \\ &= MAP(\arg\max_n \{MAP(q_{\infty,n})\}) \end{aligned} \tag{3.8}$$

where n terms are the selected terms from relevant documents set R rather than non-relevant documents set S nor user query q, and $MAP(.)$ is the function that returns the MAP value of the query in its argument (i.e., $q_{\infty,\max}$), given our baseline retrieval system VSM. However, there is more than one query with exact n terms from q_∞. Let us denote a particular asymptotic query with n terms as $q_{\infty,n,k}$ where k indicates a particular combination of n terms. Then, $q_{\infty,n}$ can be specified as:

$$q_{\infty,n} \equiv \arg\max_k \{MAP(q_{\infty,n,k})\}) \tag{3.9}$$

The number of queries that can be formulated by picking exact n terms from q_∞ is C^n_{m-n} where $card(q_\infty) = m$. Therefore, identifying $q_{\infty,n}$ is not a trivial computational problem given that n and m could be hundreds or thousands of terms.

We propose to use a greedy term selection algorithm in Figure 3.1 to formulate the asymptotic query $q_{\infty,n}$. This algorithm is based on the idea of selecting highly ranked terms. It basically ranks the terms by a term ranking function $\omega(t)$ that reflects whether the term t is useful for retrieval or not. A term is sequentially added from the ranked list of terms to $Q_{\infty,n}$ approximating the best query $q_{\infty,n}$ until $card(Q_{\infty,n}) = n$. Greedy term selection algorithm assumes that:

$$MAP(q_{\infty,n}) \approx MAP(Q_{\infty,n}) \tag{3.10}$$

which may not be entirely unrealistic because only a near best MAP performance is required and because n can be increased to a level where $MAP(Q_{\infty,n}) \approx MAP(Q_{\infty,n+1})$.

In details, our greedy term selection algorithm (Figure 3.1) uses a term ranking function $\omega(.)$ (will be defined later in Section 3.4), that indicates how good a term is for retrieval. Initially, the algorithm starts with no good terms but with a set R of relevant documents for a particular topic, the desired number n of terms in the formulated query $Q_{\infty,n}$, and an initial set q of query terms for the user bias assumption (1) (in Table 3.1). In step 2, all the terms in the relevant documents are added to the set P'. In step 3, any terms in P' are stemmed and then added to P if they are not stop words and not numerals, and they must occur in more than one document in the collection C. This is designed to avoid formulating trivial optimal queries where each relevant document is picked up by one term that only occurred

```
Method: Greedy Term Selection Algorithm
Input: set $R$, integer $n$, set $q$
Output: set $Q_{\infty,n}$

  1. set $P \leftarrow \emptyset$;
  2. set $P' \leftarrow \bigcup_{d \in R} d$;
  3. for each term $t \in P'$ do
      (a) if $t$ is a stop word then goto step 3;
      (b) $t' \leftarrow stem(t)$;
      (c) if $t' \in q$ then goto step 3;
      (d) if $t'$ is a numerical term then goto step 3;
      (e) if $(DF(C, t') < 2)$ then goto step 3;
      (f) $P \leftarrow \{t'\} \cup P$;
  4. for each term $t$ in $P$, calculate the weight by using term ranking function $\omega(t)$;
  5. rank all the terms in $P$ according to term weights in $L$;
  6. select top $n$ terms from ranked term list $L$ to formulate the output query $Q_{\infty,n}$.
  7. $Q_{\infty,n} \leftarrow q \cup Q_{\infty,n}$
```

Figure 3.1: Greedy term selection algorithm for formulating asymptotic query

in that relevant document. This is so-called near optimal query rather than optimal query in our study. In step 4, the weight of each term is calculated according to the term ranking function $\omega(.)$, that is supposed to reflect whether the term is good for retrieval or not. Then, the terms are ranked by these weights in step 5. Finally, the top n terms are selected from the ranked list. If q is the empty set, then the returned query has no user bias. If q is the initial user query, then the returned query has user bias.

3.3 Examining User Bias

In this section, we examine the impact of the user bias assumption (1) (in Table 3.1) on retrieval effectiveness. User bias refers to the original query matters. Interests in assumption (1) stem from the problem of topic lapse in practical RF [Sal71, SB90]. Apart from assumption (1), all other assumptions in Table 3.1 are assumed to hold and are studied later in this chapter.

We compare the performance of our basic IRF without the user query (i.e., q is an empty set in greedy term selection algorithm in Figure 3.1), and our basic IRF with the user query. The user queries in this experiment are the title queries (T) because these are realistic queries for many IR applications. Our VSM is used and the experiment is carried out for the TREC-6, TREC-7 and TREC-8 test collections. The term ranking function $\omega(.)$ (in Figure 3.1) in this experiment is $W4$ (i.e., $F4$) [RJ76]:

$$\begin{aligned} \omega(t) &= W4(t) \qquad (3.11) \\ &\equiv \log \frac{(DF(R,t)+0.5)\times(card(C)-DF(C,t)-card(R)+DF(R,t)+0.5)}{(DF(C,t)-DF(R,t)+0.5)\times(card(R)+DF(R,t)+0.5)} \end{aligned}$$

where $DF(C,t)$ is the document frequency of term t in the collection C, $DF(R,t)$ is the document frequency of term t in the set R of relevant documents for a particular topic. $card(.)$ is the number of documents in the given set.

In order to have more observation points, we use nine different query sizes to formulate the test queries $Q_{\infty,n}$: n =3, 25, 50, 75, 100, 125, 150, 175 and 200. The reason for selecting 3 terms as the minimum size is for comparison with the title queries (T) that usually have three terms (i.e., there are 56%, 56% and 44% title queries with 3 terms in TREC-6, TREC-7 and TREC-8 respectively). On the other hand, the maximum size of long queries (TDN) in TREC-6, TREC-7 or TREC-8 is about 150. We add two more observation points, 175 and 200, to ensure the operating range is large enough for us to decide a suitable query size for later experiments.

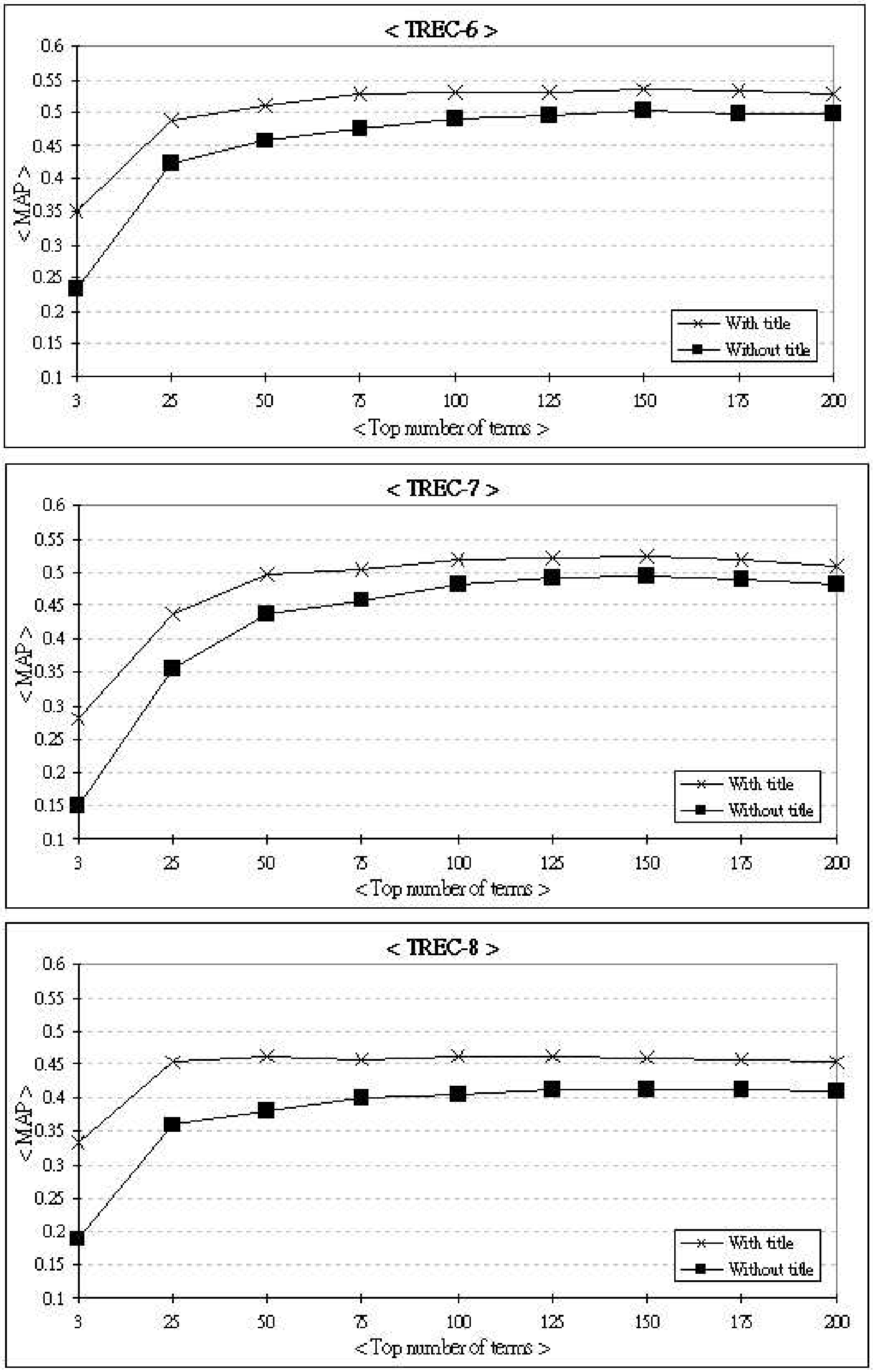


Figure 3.2: Comparing the MAP with and without user query against the different query sizes to study the user bias assumption (1)

Figure 3.2 shows the experiment results of the MAP values of the optimal queries $Q_{\infty,n}$ with user query (labeled 'With title') and without user query (labeled 'Without title') against the different query sizes. This figure shows that 'With title' achieves better MAP value than 'Without title' for all the test query sizes in all of the test collections. This better MAP value may be due to the fact that the same user formulated the query and identified the relevant document set. However, the MAP difference between 'With title' and 'Without title' diminishes as the query size increases. This suggests that the user bias assumption (1) has less and less impact on retrieval effectiveness, as expected. We use Wilcoxon 1-tailed sign test to examine whether using the title query will achieve better retrieval effectiveness or not. We choose 'top 100' queries to run this test (i.e., $n = 100$) because the MAP of these queries are close to the asymptotic performance in Figure 3.2. The Wilcoxon sign test results conclude that using title query can achieve better MAP value than using without title query with 99.7% confidence. Hence, we believe that user bias assumption (1) is valid. The title query will be added to $Q_{\infty,n}$ in all the subsequent experiments.

3.4 Dimensionality Selection - $\omega(.)$

This section examines the impact of the dimensionality reduction assumption (2) on retrieval effectiveness. Assumption (2) has been widely examined in practical RF but seldom in the IRF context. The advantage of examining assumption (2) in the IRF context is that the practical problems of RF are separated from the intrinsic problems of Assumption (2) itself. This assumption has two aspects to examine: (a) what dimensions should be selected, and (b) how many dimensions should be used. Apart from assumption (2), assumption (1) is assumed to hold by using title query, whereas assumptions (3), (4) and (5) are not asserted and are studied later.

3.4.1 What dimensions

In greedy term selection algorithm in Figure 3.1, the term ranking function $\omega(.)$ is used to determine the dimensions (i.e., query terms). In Section 3.3, we have applied a well-known function, $W4$, to rank the term, the generated queries that have achieved 0.5344, 0.522 and 0.4637 MAP values for TREC-6, 7 and 8, respectively. In this section, we mainly focus on exploring the effects of using different term ranking function $\omega(.)$. 23 basis term ranking functions are combined to yield 149 unique term ranking functions for our experiments. Some of these term ranking functions have been widely used in IR, but some have not.

The 23 basis term ranking functions can be divided into two major classes: inter-document term ranking functions $\omega_1(t)$ which is shown in Table 3.2, and intra-document term ranking functions $\omega_2(t)$ which is shown in Table 3.3. The combined term ranking function $\omega(t)$ is the product of $\omega_1(t)$ and $\omega_2(t)$, similar to the standard TF-IDF term ranking functions [SB88]:

$$\omega(t) = \omega_1(t) \times \omega_2(t) \tag{3.12}$$

Two types of inter-document term ranking functions can be distinguished: point-based ranking functions (i.e., ranking function 1 to 4 in Table 3.2); and distribution-based ranking functions (i.e., ranking function 5 to 14 in Table 3.2). For the chi-square and relative entropy ranking functions, we provide two more variations of the probability definitions other than the one in the original paper [CMRB01]: one is based on the probability of whether the term is in the document (i.e., ranking function 5, 6, 8 and 9 in Table 3.2), and the other is based on the probability that a term in the document is the desired term (i.e., ranking function 7 and 10 in Table 3.2). The intra-document term ranking functions can be divided into three types: without any normalizations (i.e., ranking function 15 to 17 in Table 3.3); with intra-document normalization (i.e., ranking function 18 to 20 in Table 3.3); and with inter- and intra-document normalization (i.e., ranking function 21 to 23 in Table

	Point-based term ranking functions
1.	$W4(t)$ as in equation 3.11
2.	$DF(R,t) = card(r \in R, t \in r)$
3.	$IDF(R,t) = log\frac{card(R)+1}{DF(R,t)}$
4.	$IDF(C,t) = log\frac{card(C)+1}{DF(C,t)}$
	Distribution-based term ranking functions
Chi-squared term ranking functions (inspired by [CMRB01])	
5.	$CHI_1(t) = \frac{[\frac{DF(R,t)}{card(R)} - \frac{DF(C,t)}{card(C)}]^2}{\frac{DF(C,t)}{card(C)}}$
6.	$CHI_2(t) = \frac{[\sum_{d \in R} \frac{TF(d,t)}{\|d\|_1} - \sum_{d \in C} \frac{TF(d,t)}{\|d\|_1}]^2}{\sum_{d \in C} \frac{TF(d,t)}{\|d\|_1}}$ as in [CMRB01]
7.	$CHI_3(t) = \frac{[\frac{1}{card(R)} \sum_{d \in R} \frac{TF(d,t)}{\|d\|_1} - \frac{1}{card(C)} \sum_{d \in C} \frac{TF(d,t)}{\|d\|_1}]^2}{\frac{1}{card(C)} \sum_{d \in C} \frac{TF(d,t)}{\|d\|_1}}$
Relative entropy term ranking functions (inspired by [CMRB01])	
8.	$KLD_1(t) = \frac{DF(R,t)}{card(R)} \times log_{10} \frac{DF(R,t) \times card(C)}{DF(C,t) \times cardR}$
9.	$KLD_2(t) = \sum_{d \in R} \frac{TF(d,t)}{\|d\|_1} \times log_{10} \frac{\sum_{d \in R} \frac{TF(d,t)}{\|d\|_1}}{\sum_{d \in C} \frac{TF(d,t)}{\|d\|_1}}$ as in [CMRB01]
10.	$KLD_3(t) = \frac{1}{card(R)} \sum_{d \in R} \frac{TF(d,t)}{\|d\|_1} \times log_{10} \frac{card(C) \sum_{d \in R} \frac{TF(d,t)}{\|d\|_1}}{card(R \sum_{d \in C} \frac{TF(d,t)}{\|d\|_1}}$
Robertson Selection Value(RSV) by Robertson et al. [RJ76]	
11.	$RSV_1(t) = W4(t) \times (\frac{DF(R,t)}{card(R)} - 0.15 \times \frac{DF(C,t)-DF(R,t)}{card(C)-card(R)})$ as used in Okapi at TREC-6 & 7
12.	$RSV_2(t) = W4(t) \times (\frac{DF(R,t)}{card(R)} - \frac{DF(C,t)-DF(R,t)}{card(C)-card(R)})$ as in [Eft93]
13.	$RSV_3(t) = DF(R,t) \times IDF(C,t) - log \begin{pmatrix} card(R) \\ DF(R,t) \end{pmatrix} - log \sum_{d \in C} \|d\|_1$ as used in Okapi at TREC-8
Expected Mutual Information Measure (EMIM) by [RHP81]	
14.	$EMIM(t) = log_{10} \frac{DF(R,t) \times card(C)}{card(R) \times DF(C,t)} \times DF(R,t) - log_{10} \frac{(DF(C,t)-DF(R,t)) \times card(C)}{(card(C)-card(R)) \times DF(C,t)}$ $\times (DF(C,t) - DF(R,t)) + log_{10} \frac{(card(C)-DF(C,t)-card(R)+DF(R,t)) \times card(C)}{(card(C)-DF(C,t)) \times (card(C)-card(R))}$ $\times (card(C) - DF(C,t) - card(R) + DF(R,t))$ $-log_{10} \frac{(card(R)-DF(R,t)) \times card(C)}{(card(C)-DF(C,t)) \times card(R)} \times (card(R) - DF(R,t))$

Table 3.2: Inter-document basis term ranking functions $\omega_1(t)$

No normalization	
15.	$MaxTF(R,t) = \max_{d \in R} TF(d,t)$
16.	$MinTF(R,t) = \min_{d \in R} TF(d,t)$
17.	$SumTF(R,t) = \sum_{d \in R} TF(d,t)$
Intra-document normalization	
18.	$MaxNTF(R,t) = \max_{d \in R} \frac{TF(d,t)}{\|d\|_1}$
19.	$MinNTF(R,t) = \min_{d \in R} \frac{TF(d,t)}{\|d\|_1}$
20.	$SumNTF(R,t) = \sum_{d \in R} \frac{TF(d,t)}{\|d\|_1}$
Fully normalization	
21.	$NMaxNTF(t) = \frac{MaxNTF(t)}{\max_{d \in C} \frac{TF(d,t)}{\|d\|_1}}$
22.	$NMinNTF(t) = \frac{MinNTF(t)}{\min_{d \in C} \frac{TF(d,t)}{\|d\|_1}}$
23.	$NSumNTF(t) = \frac{SumNTF(t)}{\sum_{d \in C} \frac{TF(d,t)}{\|d\|_1}}$

Table 3.3: Intra-document basis term ranking functions $\omega_2(t)$

3.3). Each type has three term ranking functions: one that is based on the maximum term frequency, or the minimum term frequency or the total term frequency of a term in the documents.

In this experiment, we use greedy term selection algorithm in Figure 3.1 to formulate the asymptotic query, except that step 4 has been modified to cater for a second term ranking function because many terms have tied term ranking function scores after visual inspection. We use $W4(.)$ as the second term ranking function for all the term ranking functions except $W4(.)$ itself. For $W4(.)$, $SumNTF(.)$ is the second term ranking function. In addition, the number of terms n appended on the title queries is 100.

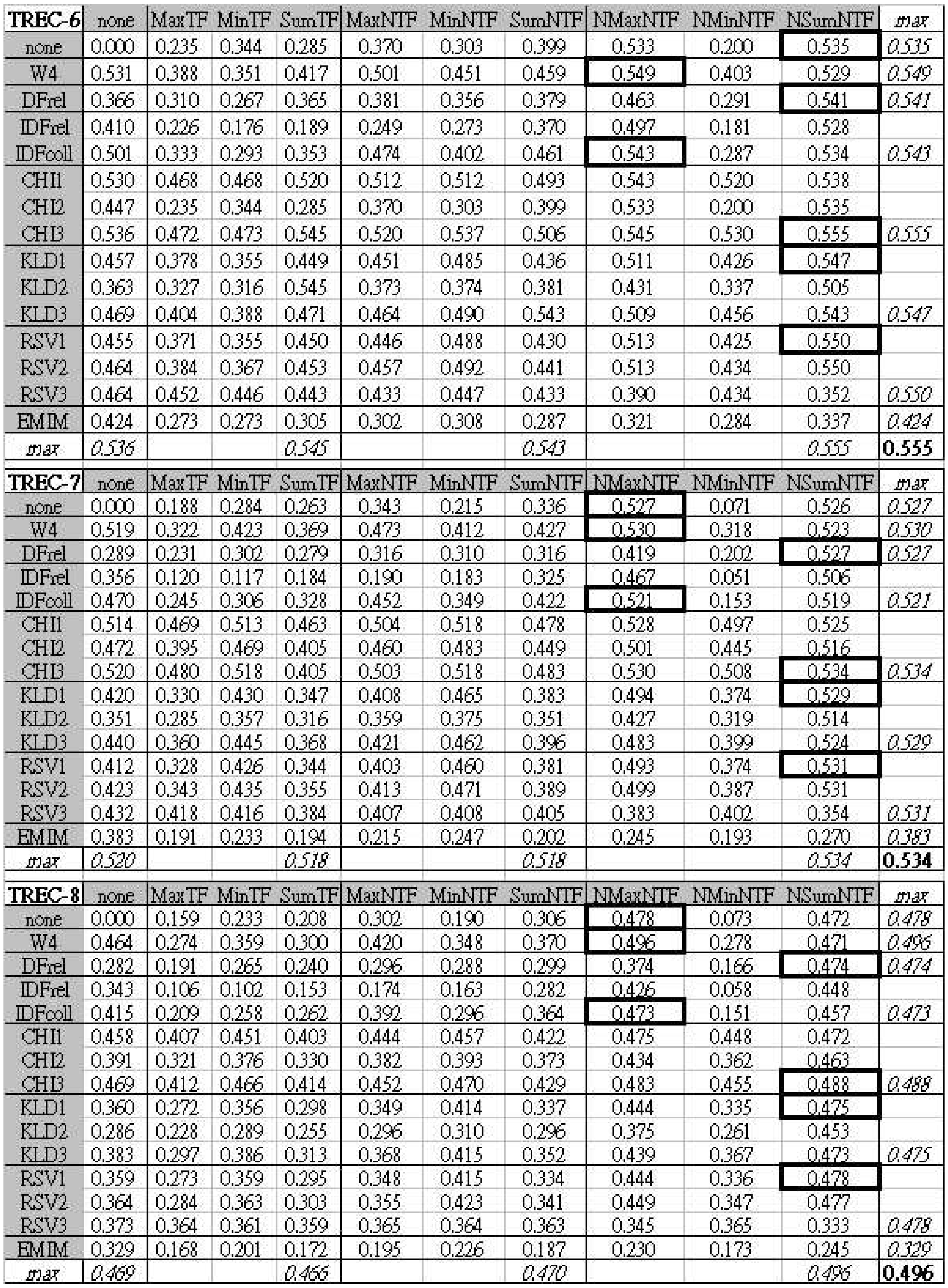
The MAP values obtained using different term ranking functions are shown in Figure 3.3. We have observed many interesting results. First, the term ranking function (7) using CHI_3 marginally achieved the best MAP value in TREC-6, 7 and 8 without any intra-document ranking function (i.e., labeled 'none' in Figure 3.3 and $\omega_2(.) = 1$). Second, in terms of multiplying intra-document ranking function $\omega_2(.)$ without normalization, $MaxTF$ and $MinTF$, actually degraded the MAP values (i.e., worse than with $\omega_2(.) = 1$), except the CHI_3, KLD_2 and KLD_3 with $SumTF$ in TREC-6 achieved better results. This implies that adding raw term frequency may not be a good method to improve the retrieval effectiveness. Third, better MAP values were obtained by multiplying $\omega_2(.)$ with intra-document normalized term frequencies (i.e., $MaxNTF$, $MinNTF$ and $SumNTF$) than without normalized term frequencies (i.e., $MaxTF$, $MinTF$ and $SumTF$). Fourth, $NMaxNTF$ and $NSumNTF$, which normalized the term frequency with respect to the document and the entire collection, achieved better MAP values than others except $EMIM$. Fifth, the results reveal that using minimum term frequency (normalized or not) do not enhance the retrieval effectiveness compared with that without intra-document term ranking. Sixth, CHI_3 achieved better MAP value than CHI_1 and CHI_2. Similarly, KLD_2 achieved the worst performance among other KLD term ranking functions. This implies that macro-average of the normalized

	$\omega_2(t)$					
	$NMaxNTF$			$NSumNTF$		
$\omega_1(t)$	TREC-6	TREC-7	TREC-8	TREC-6	TREC-7	TREC-8
Unity	-	0.527	0.478	0.533	-	-
$W4$	0.549	0.530	**<u>0.496</u>**	-	-	-
DF	-	-	-	0.541	0.527	0.474
IDF	0.543	0.521	0.473	-	-	-
CHI_3	-	-	-	**<u>0.555</u>**	**<u>0.534</u>**	0.488
KLD_1	-	-	-	0.547	0.529	0.475
RSV_1	-	-	-	0.550	0.531	0.478

Table 3.4: Summary of the performance for the better term ranking functions

probability of term occurrence is a more appropriate probability function for term rankings. Seventh, RSV_1 is the best among other RSV term ranking functions. Eighth, RSV_3 and $EMIM$ perform worst than others. Finally, the common inverse document frequency (i.e., IDF), has a MAP better than 0.5, except for TREC-8. Therefore, existing IDF function is already effective in selecting good terms for RF.

Table 3.4 summarizes the retrieval effectiveness of the better term ranking functions which are highlighted with rectangles in Figure 3.3. We can see that the $\omega_2(.)$ of the better ranking functions is either the fully normalized maximum term frequency or the fully normalized summed term frequency. Moreover, it seems that the combination of a particular $\omega_1(.)$ with another $\omega_2(.)$ that produces the best result is stable across different TREC test collections, except when $\omega_1(.)$ is unity. Furthermore, the combination of CHI_3 with the fully normalized summed term frequency ($CHI_3 \times NSumNTF$) produces the overall best results for TREC-6 and TREC-7 test collections, and $W4$ combined with the fully normalized maximum term frequency ($W4 \times NMaxNTF$) produces the overall best results for TREC-8. However, these overall best results are not statistically significantly different from the results of other better term ranking functions in Table 3.4.



TREC-6	none	MaxTF	MinTF	SumTF	MaxNTF	MinNTF	SumNTF	NMaxNTF	NMinNTF	NSumNTF	*max*
none	0.000	0.235	0.344	0.285	0.370	0.303	0.399	0.533	0.200	$\boxed{0.535}$	*0.535*
W4	0.531	0.388	0.351	0.417	0.501	0.451	0.459	$\boxed{0.549}$	0.403	0.529	*0.549*
DFrel	0.366	0.310	0.267	0.365	0.381	0.356	0.379	0.463	0.291	$\boxed{0.541}$	*0.541*
IDFrel	0.410	0.226	0.176	0.189	0.249	0.273	0.370	0.497	0.181	0.528	
IDFcoll	0.501	0.333	0.293	0.353	0.474	0.402	0.461	$\boxed{0.543}$	0.287	0.534	*0.543*
CHI1	0.530	0.468	0.468	0.520	0.512	0.512	0.493	0.543	0.520	0.538	
CHI2	0.447	0.235	0.344	0.285	0.370	0.303	0.399	0.533	0.200	0.535	
CHI3	0.536	0.472	0.473	0.545	0.520	0.537	0.506	0.545	0.530	$\boxed{0.555}$	*0.555*
KLD1	0.457	0.378	0.355	0.449	0.451	0.485	0.436	0.511	0.426	$\boxed{0.547}$	
KLD2	0.363	0.327	0.316	0.545	0.373	0.374	0.381	0.431	0.337	0.505	
KLD3	0.469	0.404	0.388	0.471	0.464	0.490	0.543	0.509	0.456	0.543	*0.547*
RSV1	0.455	0.371	0.355	0.450	0.446	0.488	0.430	0.513	0.425	$\boxed{0.550}$	
RSV2	0.464	0.384	0.367	0.453	0.457	0.492	0.441	0.513	0.434	0.550	
RSV3	0.464	0.452	0.446	0.443	0.433	0.447	0.433	0.390	0.434	0.352	*0.550*
EMIM	0.424	0.273	0.273	0.305	0.302	0.308	0.287	0.321	0.284	0.337	*0.424*
max	*0.536*			*0.545*			*0.543*			*0.555*	**0.555**

TREC-7	none	MaxTF	MinTF	SumTF	MaxNTF	MinNTF	SumNTF	NMaxNTF	NMinNTF	NSumNTF	*max*
none	0.000	0.188	0.284	0.263	0.343	0.215	0.336	$\boxed{0.527}$	0.071	0.526	*0.527*
W4	0.519	0.322	0.423	0.369	0.473	0.412	0.427	$\boxed{0.530}$	0.318	0.523	*0.530*
DFrel	0.289	0.231	0.302	0.279	0.316	0.310	0.316	0.419	0.202	$\boxed{0.527}$	*0.527*
IDFrel	0.356	0.120	0.117	0.184	0.190	0.183	0.325	0.467	0.051	0.506	
IDFcoll	0.470	0.245	0.306	0.328	0.452	0.349	0.422	$\boxed{0.521}$	0.153	0.519	*0.521*
CHI1	0.514	0.469	0.513	0.463	0.504	0.518	0.478	0.528	0.497	0.525	
CHI2	0.472	0.395	0.469	0.405	0.460	0.483	0.449	0.501	0.445	0.516	
CHI3	0.520	0.480	0.518	0.405	0.503	0.518	0.483	0.530	0.508	$\boxed{0.534}$	*0.534*
KLD1	0.420	0.330	0.430	0.347	0.408	0.465	0.383	0.494	0.374	$\boxed{0.529}$	
KLD2	0.351	0.285	0.357	0.316	0.359	0.375	0.351	0.427	0.319	0.514	
KLD3	0.440	0.360	0.445	0.368	0.421	0.462	0.396	0.483	0.399	0.524	*0.529*
RSV1	0.412	0.328	0.426	0.344	0.403	0.460	0.381	0.493	0.374	$\boxed{0.531}$	
RSV2	0.423	0.343	0.435	0.355	0.413	0.471	0.389	0.499	0.387	0.531	
RSV3	0.432	0.418	0.416	0.384	0.407	0.408	0.405	0.383	0.402	0.354	*0.531*
EMIM	0.383	0.191	0.233	0.194	0.215	0.247	0.202	0.245	0.193	0.270	*0.383*
max	*0.520*			*0.518*			*0.518*			*0.534*	**0.534**

TREC-8	none	MaxTF	MinTF	SumTF	MaxNTF	MinNTF	SumNTF	NMaxNTF	NMinNTF	NSumNTF	*max*
none	0.000	0.159	0.233	0.208	0.302	0.190	0.306	$\boxed{0.478}$	0.073	0.472	*0.478*
W4	0.464	0.274	0.359	0.300	0.420	0.348	0.370	$\boxed{0.496}$	0.278	0.471	*0.496*
DFrel	0.282	0.191	0.265	0.240	0.296	0.288	0.299	0.374	0.166	$\boxed{0.474}$	*0.474*
IDFrel	0.343	0.106	0.102	0.153	0.174	0.163	0.282	0.426	0.058	0.448	
IDFcoll	0.415	0.209	0.258	0.262	0.392	0.296	0.364	$\boxed{0.473}$	0.151	0.457	*0.473*
CHI1	0.458	0.407	0.451	0.403	0.444	0.457	0.422	0.475	0.448	0.472	
CHI2	0.391	0.321	0.376	0.330	0.382	0.393	0.373	0.434	0.362	0.463	
CHI3	0.469	0.412	0.466	0.414	0.452	0.470	0.429	0.483	0.455	$\boxed{0.488}$	*0.488*
KLD1	0.360	0.272	0.356	0.298	0.349	0.414	0.337	0.444	0.335	$\boxed{0.475}$	
KLD2	0.286	0.228	0.289	0.255	0.296	0.310	0.296	0.375	0.261	0.453	
KLD3	0.383	0.297	0.386	0.313	0.368	0.415	0.352	0.439	0.367	0.473	*0.475*
RSV1	0.359	0.273	0.359	0.295	0.348	0.415	0.334	0.444	0.336	$\boxed{0.478}$	
RSV2	0.364	0.284	0.363	0.303	0.355	0.423	0.341	0.449	0.347	0.477	
RSV3	0.373	0.364	0.361	0.359	0.365	0.364	0.363	0.345	0.365	0.333	*0.478*
EMIM	0.329	0.168	0.201	0.172	0.195	0.226	0.187	0.230	0.173	0.245	*0.329*
max	*0.469*			*0.466*			*0.470*			*0.496*	**0.496**

Figure 3.3: The MAP of 149 term ranking functions $\omega(.)$

3.4.2 How many dimensions

The other aspect of the dimension reduction assumption (2) is how many dimensions should be used, which is equal to deciding the number of query terms n in asymptotic query q_∞. In the previous experiment of evaluating different term ranking function, the number of appended terms is fixed to top 100 to eliminate the effects of query size. In this experiment, on the other hand, we pick the seven better term ranking functions in Table 3.4 and vary the query size from 0 to 200: 0, 3, 25, 50, 75, 100, 125, 150, 175 and 200. The experimental results can be used to examine the impacts of different query sizes and to determine if any better retrieval effectiveness can be obtained.

Figure 3.4 shows the experimental results. The MAP values of all the seven term ranking functions do not change much when the query with size of 100 or larger (i.e., Top 100 terms). These experimental results are then analyzed with the ANOVA test to determine whether the changes are significance. According to the ANOVA test with 95% confidence, queries with size of 100 have significant improvement in MAP compared with those query sizes smaller than 100 (i.e., from size 0 to 75); however, no significant improvement can be gained when comparing with those query sizes larger than 100 (i.e., from size 125 to 200). Therefore, we believe that query size of 100 is already the (near) best.

Nevertheless, many researchers [BH03] have illustrated that the sizes of the optimal query of different topics are different. It means that we cannot use the same query size for all the topics. In order to investigate this effect, we compare the MAP of using best query size in each topic and the MAP of using same query size 100 for all topics with two of our best term ranking functions, $CHI_3 \times NSumNTF$ and $W4 \times NMaxNTF$. From Table 3.5 we can see that using best query size in each topic is always better than using same query size 100. However, according to the ANOVA test with 95% confidence, these improvements are not significant.

If we use the marginally better term ranking function $CHI_3 \times NSumNTF$

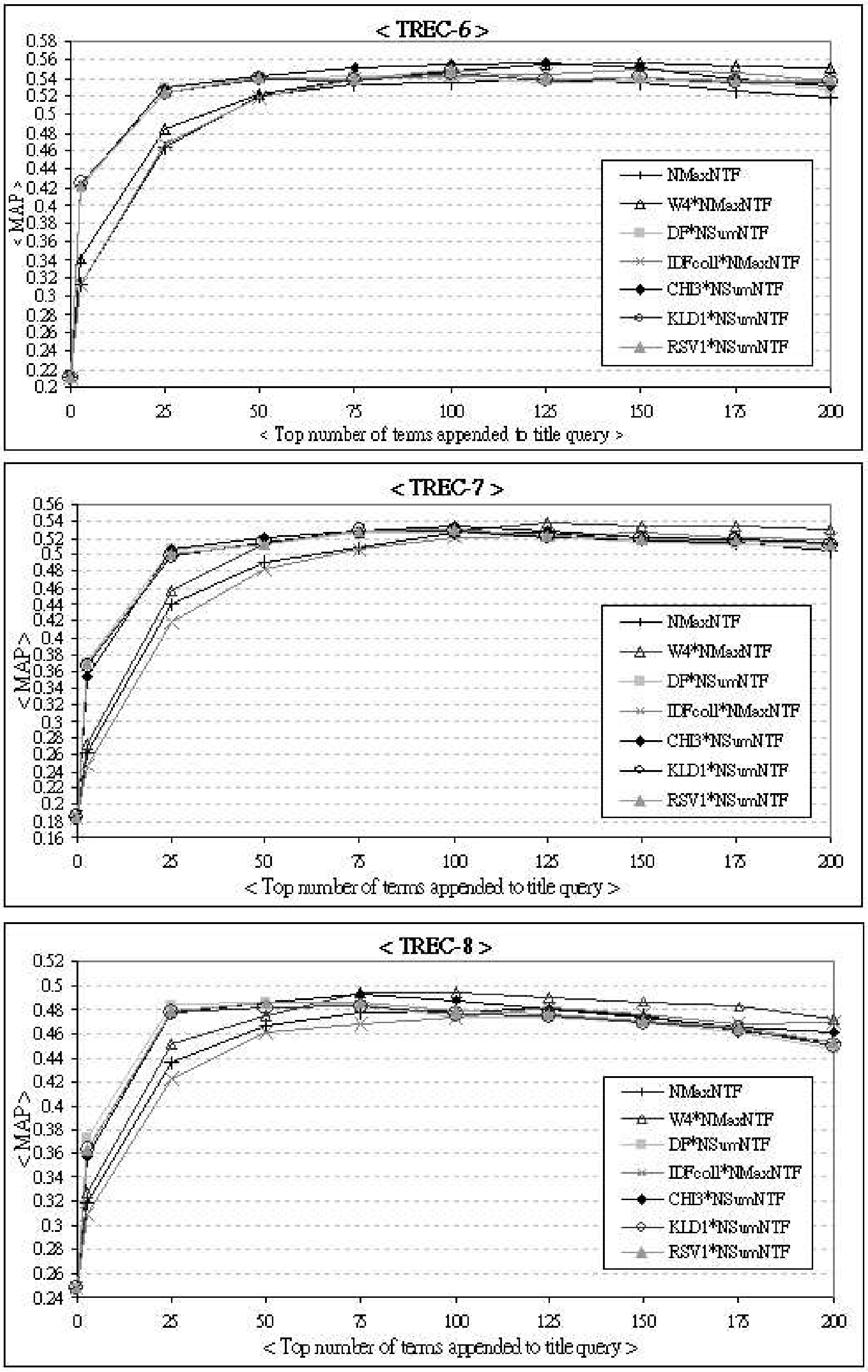


Figure 3.4: The MAP of 7 better term ranking functions $\omega(.)$ against the different number of top ranked terms

	Term rank function $\omega(t)$	Top 100	Best size	BestSize>Top100 (95% conf.)
TREC-6	$CHI_3 \times NSumNTF$	0.5546	0.6162	No
	$W4 \times NMaxNTF$	0.5487	0.6098	No
TREC-7	$CHI_3 \times NSumNTF$	0.5343	0.5917	No
	$W4 \times NMaxNTF$	0.5298	0.5941	No
TREC-8	$CHI_3 \times NSumNTF$	0.4884	0.5557	No
	$W4 \times NMaxNTF$	0.4946	0.5574	No

Table 3.5: The MAP of query with size of 100 and the best query size in each topic

	Retrieval Mode	Known Best	IRF	IRF>KnownBest (95% conf.)
TREC-6	Manual(RF)	0.463	0.616	Yes
	Automatic	0.300		Yes
TREC-7	Manual(RF)	0.370	0.592	Yes
	Automatic	0.303		Yes
TREC-8	Manual(RF)	0.469	0.556	No
	Automatic	0.327		Yes

Table 3.6: The MAP of our IRF and the best known ad hoc retrieval results in TREC

with the best query size in each topic, the average retrieval effectiveness of our IRF for TREC-6, 7 and 8 can be improved to $MAP_o = 0.6162$, $MAP_o = 0.5917$ and $MAP_o = 0.5557$, respectively. It should be noted that these results are better than those best in TREC ad hoc retrieval (as shown in Table 3.6), and the improvements are statistically significant with 95% confidence. Therefore, we believe that our IRF may be useful as a mechanism to determine the performance limits of retrieval systems in an open evaluation workshop, such as TREC or NTCIR. Furthermore, we can conclude that there is still rooms to improve the current RF.

3.5 Asymptotic performance

In this section, we explore the asymptotic assumption (3) by using different numbers of top ranked retrieved documents from the initial retrieval list. In general, as the number of RF iteration m increases, the retrieved and relevant documents U_m get larger, and the corresponding recall of retrieval increases. As m tends to infinity, U_∞ tends to R. In practice, this increase of the size of U_m can be simulated by getting different numbers of top j ranked retrieved documents from the initial retrieval list. Therefore, we can plot the performance against the number of top retrieved documents and the recall which would represent the trend of the IRF. This experiment also can illustrate the issue in practical RF and PRF that where to select feedback terms is better.

The process of this experiment is similar to PRF but has an idealized classifier that identifies all relevant documents from top retrieved relevant documents without errors (this is called IPRF in our study). Firstly, an initial query q is used to retrieve the initial ranked retrieval list. Then, the top j retrieved document is classified into relevant documents U_j and non-relevant documents V_j by the idealized classifier. Finally, the query q_j is formulated by Equation 3.13. This equation is similar to Equation 3.7 but it discards the asymptotic assumption (3). It is because we are interested in investigating how many top j retrieved documents in practical RF can formulate the asymptotic query q_∞ (i.e., approximate optimal query q_o):

$$q_j = q \cup \bigcup_{d' \in U_j} d' \tag{3.13}$$

Two types of user queries, title (T) and long (TDN), are used to examine the impact of the initial query q as well as initial retrieval list on performance. The MAP values obtained by select terms from the relevant documents in the top 10, 20, 40, 80, 160, 320, 640, 1000, 10000, possibly 100000 and the entire retrieved list are plotted for visualization. The seven better term ranking functions listed in Table 3.4 are used to formulate the query q_j with size of 100. In Figure 3.5, we plotted the IRF performance at 556077, 528155 and 528155, which is the number of documents in TREC-6, 7 and 8 collections, respectively. Moreover, the average number of retrieved documents in the entire retrieval list for T initial query is equal to 59387, 35781 and 42395 for TREC-6, 7 and 8, respectively, and for TDN initial query is equal to 409,902, 305,827 and 297,049 for TREC-6, 7 and 8, respectively, which is close to the size of each collection. In Figure 3.6, we plot the mean recall of each query against the MAP, and the IRF is plotted with a recall of 100%. The numbers of top retrieved documents are labeled near the data points in both figures.

Figure 3.5 clearly shows that the MAP values of different top j retrieved documents are lower than IRF. This implies that many practical RF will achieve a performance lower than IRF as the user examines a limited number of top j retrieved documents in practice. This serves to confirm our IRF Hypothesis (1). Moreover, Figures 3.5 and 3.6 illustrate that using more top retrieved documents achieves better MAP and mean recall for all kinds of initial queries and term ranking functions. This implies that if the user examines more top j retrieved documents, the retrieval effectiveness of practical RF will be improved. However, Figure 3.6 suggests that if the practical RF (i.e., from top 10 retrieved documents to the entire retrieved list in Figure 3.6) cannot achieve near 100% recall, then it is highly unlikely that the MAP of the practical RF can approach the MAP performance of its IRF. So it is not necessary to examine the whole retrieval list in practice. From Figure 3.5 we can see that the retrieval effectiveness is increased sharply until the top 100. Therefore, we believe that selected feedback terms from top 100 retrieved documents are effective in RF as well as PRF. Furthermore, for long queries, using the entire the retrieval list can achieve similar performance to the IRF. This implies that we can simulate our IRF by using the entire relevant retrieval list. For title queries, the MAP using the entire retrieval list is about 5% lower than that of the IRF version. This suggests that for title queries RF needs at least two iterations in order to ensure that the retrieval list has an adequately high recall to obtain the potential IRF performance.

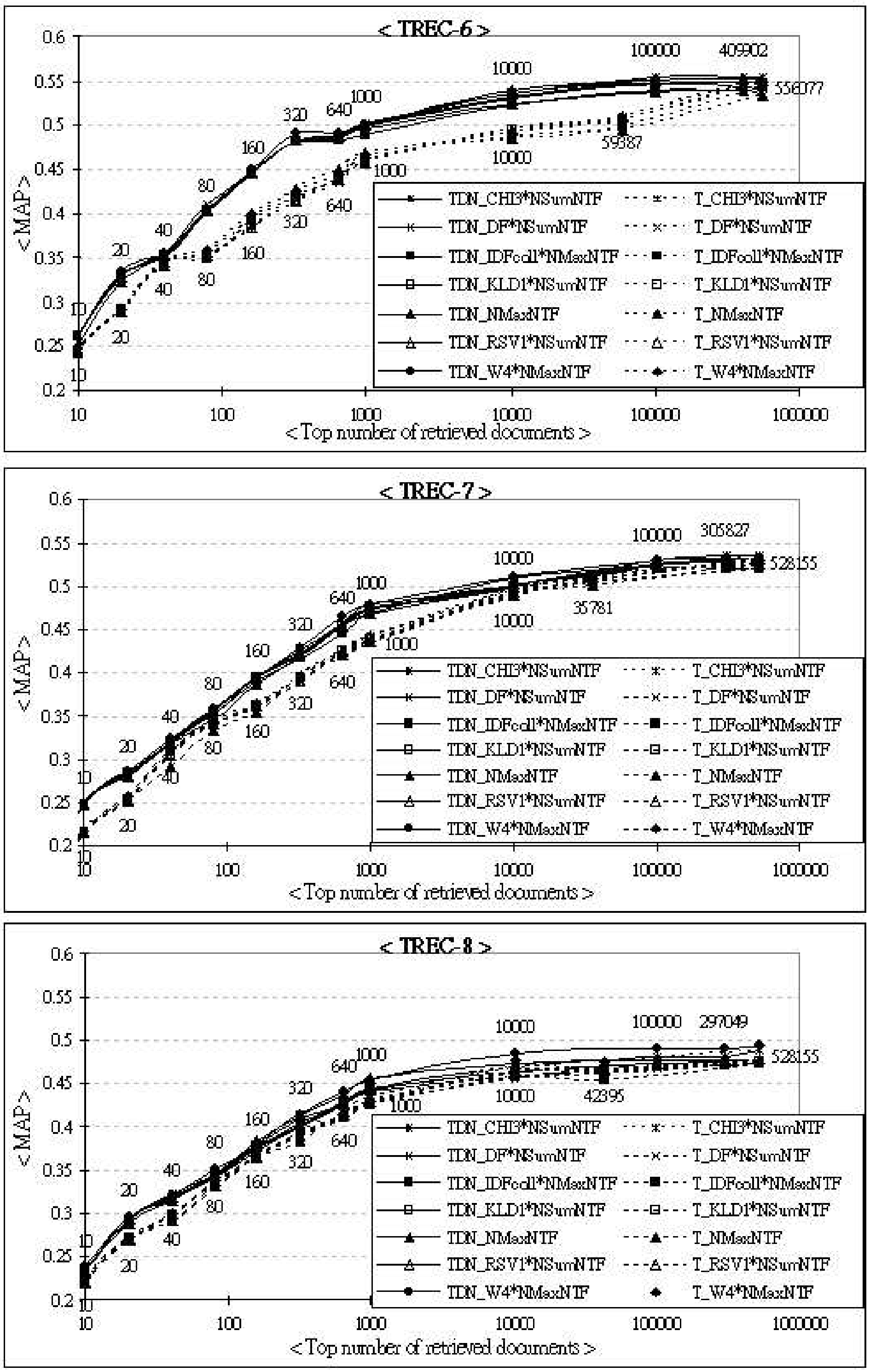
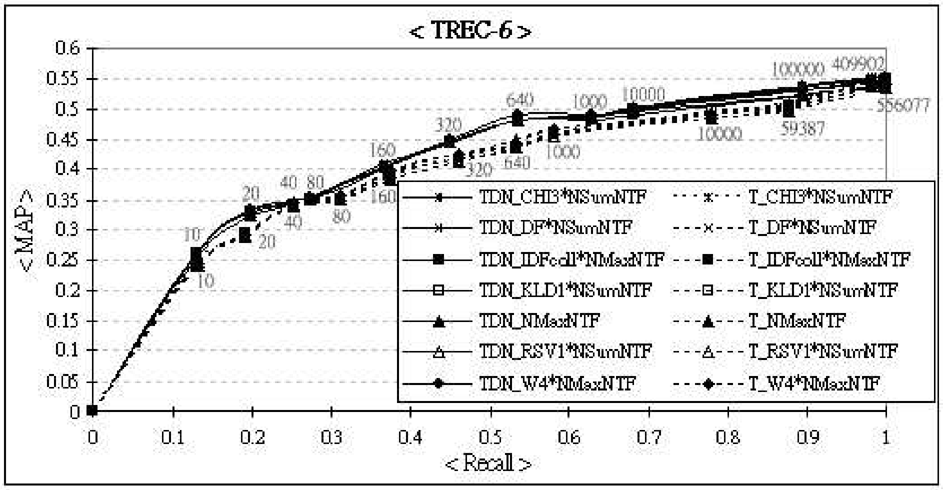
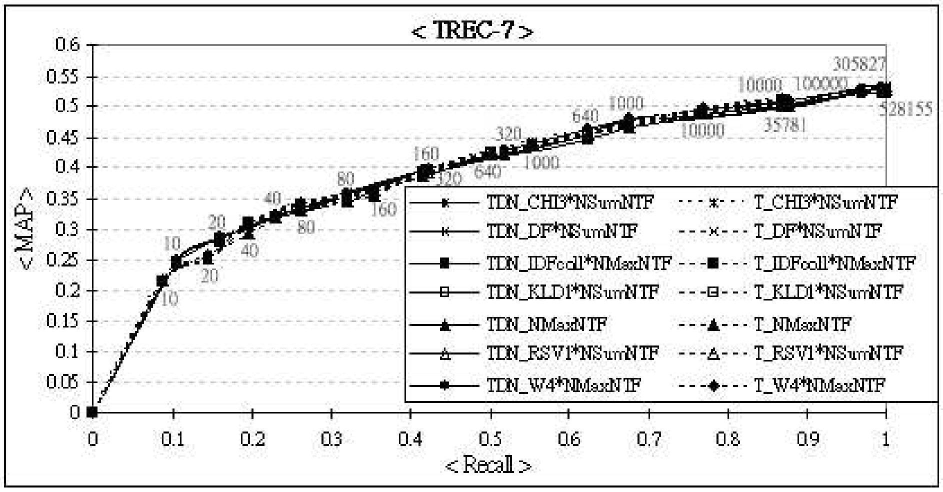


Figure 3.5: The MAP of different term ranking functions against the top number of retrieved documents for long (TDN: in solid lines) and title (T: in dotted lines) queries.





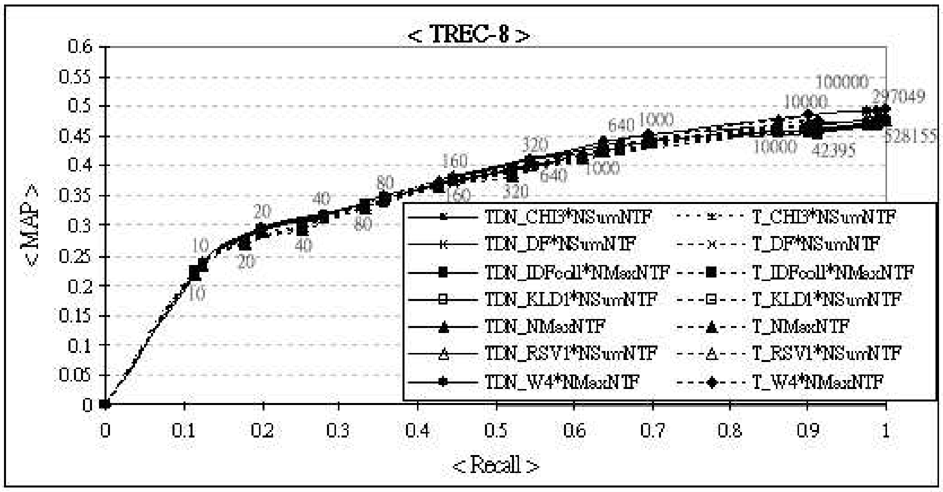


Figure 3.6: The MAP of different term ranking functions against the mean recall for long (TDN: in solid lines) and title (T: in dotted lines) queries.

3.6 Positively Weighted Terms

This section examines the unweighted assumption (4) that allows us to assign weights to query terms instead of using unity weights for all query terms. However, only positive weights are considered (i.e., $\gamma = 0$), and negative weights will be examined in the next section. Three other assumptions hold in the case. The asymptotic query q_∞ is simply modified from Equation 3.5 to the Equation 3.14 instead, and step 7 of greedy term selection algorithm in Figure 3.1 is modified to $Q_{\infty,n} \leftarrow \alpha \sum_{t \in q}(\sigma_q(t) \times t) + \beta \sum_{t \in Q_{\infty,n}}(\sigma_f(t) \times t)$, where $\sigma_q(t)$ and $\sigma_f(t)$ are the term weight functions of user query q and feedback terms $Q_{\infty,n}$, respectively, these are used to assign the weight to term t. In this experiment, the marginally best term ranking function (i.e., $CHI_3 \times NSumNTF$), is used to formulate the query with size of 100. The weights are mixed using the parameter α and $\beta(= 1 - \alpha)$:

$$q_\infty = \alpha q + \frac{\beta}{card(R)} \sum_{d' \in R} d' \tag{3.14}$$

Five term weight normalization schemes are proposed in Table 3.7. First, Roochio positive scheme (1) is the standard Rocchio relevance feedback, except that there are no negatively weighted terms. The weights of user query q are calculated by the inverse of the number of terms in the user query. The weights of feedback terms are calculated by the term ranking function $\omega(.)$, which is $CHI_3 \times NSumNTF$ in this experiment. Second, Ide-Regular scheme (2) [IS71] is similar to scheme (1) but the term weights are linearly mixed as shown in Equation 3.15:

$$q_\infty = \alpha q + \beta \sum_{d' \in R} d' \tag{3.15}$$

Third, city-block normalization scheme (3) can be considered as the scheme (2) where the feedback term weights are normalized by the sum of its term weight. Fourth, rank normalization scheme (4) can also be considered as the scheme (2) where the term weight of term t is normalized by its rank, $rank(t)$. This rank is

No.	Scheme Name	User query weights $\sigma_q(t)$	Feedback terms weights $\sigma_f(t)$	Asymptotic query formula q_∞
1	Rocchio Positive	$\frac{1}{\lvert q \rvert_{term}}$	$\omega(t)$	as in Equation 3.14
2	Ide-Regular	$\frac{1}{\lvert q \rvert_{term}}$	$\omega(t)$	as in Equation 3.15
3	City-block Normalization	$\frac{1}{\lvert q \rvert_{term}}$	$\frac{\omega(t)}{\sum_{t \in Q_{\infty,n}} \omega(t)}$	as in Equation 3.15
4	Rank Normalization	$\frac{1}{\lvert q \rvert_{term}}$	$\frac{n - rank(t) + 1}{\sum_{t \in Q_{\infty,n}} rank(t)}$	as in Equation 3.15
5	Unity Weight	1	1	as in Equation 3.15

Table 3.7: Different term weight normalization schemes

determined by the term ranking function: $CHI_3 \times NSsumNTF$ in this experiment. The last scheme, unity weight scheme (5), is essentially without any weighting since every term is as important as the other.

Figure 3.7 shows the MAP values of the queries using five different term weight normalization schemes and different parameter values α and β. It is obvious that the MAP values of the Ide-regular scheme (2) and unity weight scheme (5) are fairly independent of the parameter values, whereas the other three schemes are sensitive to different parameter values. Moreover, only the rank normalization scheme (4) with specific parameter values (i.e., when $\alpha = 0.1$) achieves higher MAP value than the unity weight scheme (5). Therefore, we only choose schemes (4) and (5) to further examine whether the rank normalization scheme (4) can perform better if the number of terms selected is different. In addition, when user query is ignored (i.e., when $\alpha = 0$), most of the schemes achieve lower MAP value than those with user query (i.e., when α =0.1 to 0.9, β =0.9 to 0.1) except Rocchio positive scheme (1). This indicates that the user bias assumption (1) is valid in either unweighted query or weighted query.

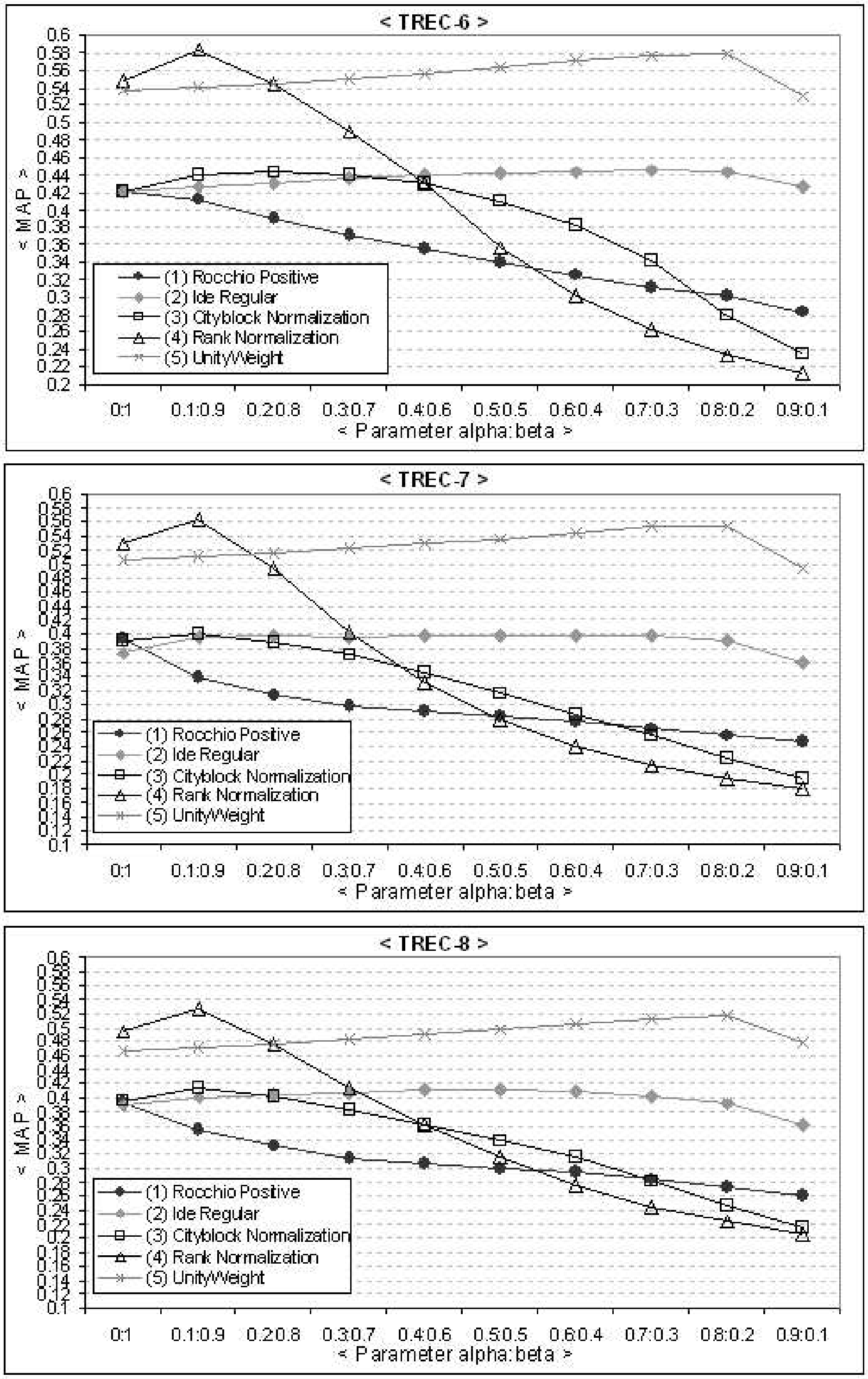


Figure 3.7: The MAP of different term weight normalization schemes against the different parameter values α:β

	Top 175	Best size	Best Size>Top 175 (95% conf.)
TREC-6	0.5972	0.6402	No
TREC-7	0.5684	0.5985	No
TREC-8	0.5351	0.5743	No

Table 3.8: The MAP of top 175 positive terms and the best query size in each topic using the best term weight normalization scheme

In order to determine the suitable query size for positively weighted terms, we picked the best two term weight normalization schemes, the rank normalization scheme (4) with $\alpha = 0.1$ and the unity weight scheme (5) with $\alpha = 0.8$, and vary the query size from 0 to 300 to examine the effects. The reason for adding four more observation points (i.e., 225, 250, 275 and 300) is because the MAP values are continuously increasing from query size of 0 to 200. These points are used to determine if any better retrieval effectiveness can be obtained. The experimental results are shown in Figure 3.8. From the figure we can see that the MAP values of the scheme (4) are always slightly better than scheme (5). The best MAP value of the scheme (4) is obtained when the query size is about 175.

Table 3.8 shows the MAP value of rank normalization scheme (4) with top 175 positive terms and with the best query size in each topic. The MAP value of the best query size is better. However, according to the result of the ANOVA test with 95% confidence, these improvements are not statistically significant. Hence, we conclude that the MAP values may not be improved by using various query sizes for different topics in positively weighted terms.

Comparing the best MAP value of the queries using positively weighted terms (the results in Table 3.8) and unweighted terms (the results in Table 3.5) for fixed query size, using positively weighted terms is about 4% higher than using unweighted terms. However, from Table 3.9 we can see that this difference in MAP value is not statistically significant [VB02]. Therefore, we conclude that positively weighted terms can only slightly enhance the retrieval effectiveness.

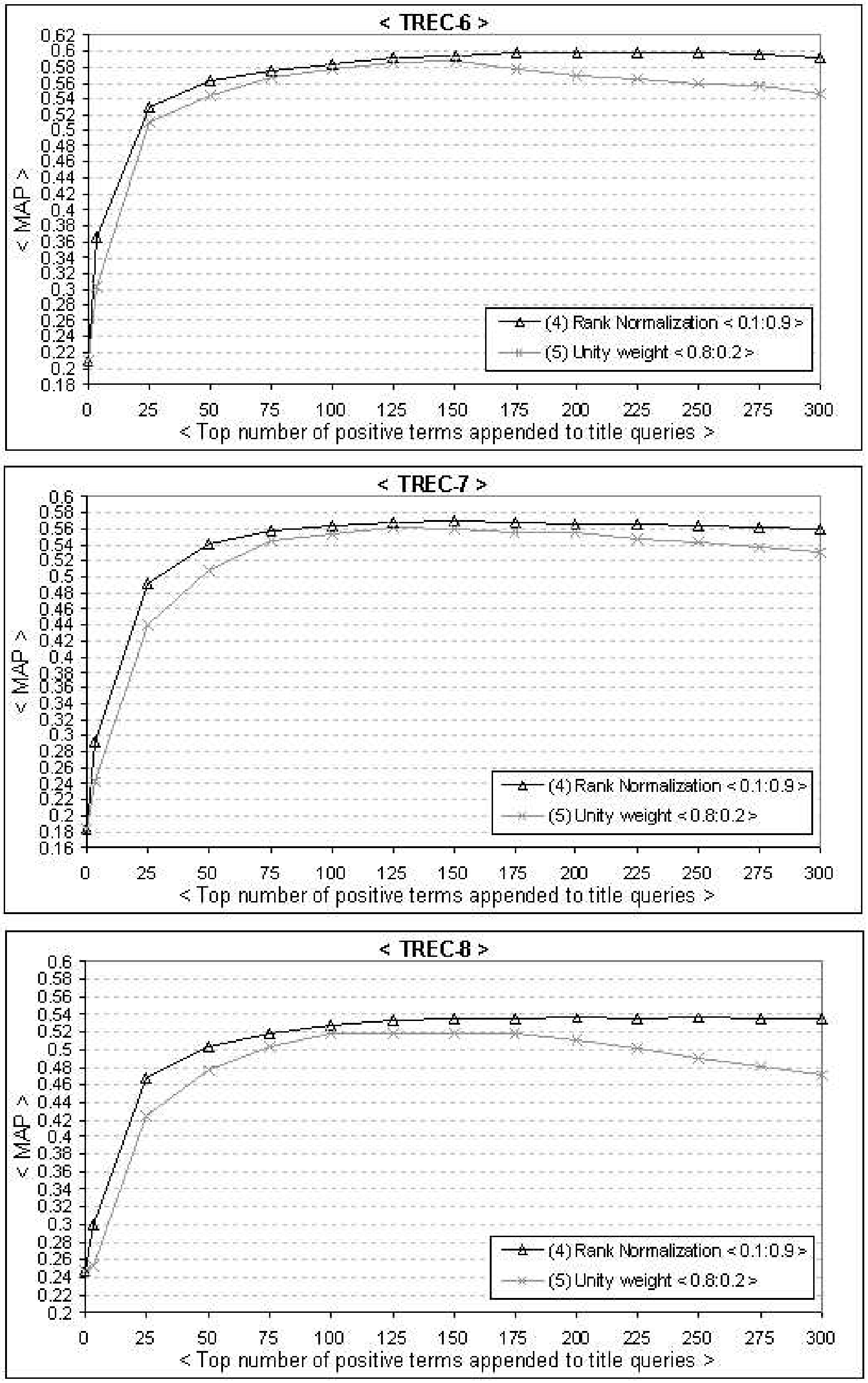


Figure 3.8: The MAP of two better term weight normalization schemes against the different number of top ranked positive terms

	Unweighted (Un)	Positively Weighted (Pos)	Pos $>$ Un (95% conf.)
TREC-6	0.555	0.597	No
TREC-7	0.534	0.568	No
TREC-8	0.495	0.535	No

Table 3.9: The MAP of our IRF with Unweighted Terms and Positively Weighted Terms with fixed query size

3.7 Negatively Weighted Terms

In this section, we examine the impact of non-negative weight assumption (5) on retrieval effectiveness. This assumption adds terms with negative weights. In the literature, negative weights did not always have a positive impact on performance [Ide71, Dun97]. Here, the assumptions (1), (2) and (3) are held and the asymptotic query q_∞ is similar to Equation 3.5, but uses a subset V_j of the top j retrieved and non-relevant documents of the user query rather than the whole set S of non-relevant documents for the topic. The reason for using the top j retrieved documents is because the whole set S of non-relevant documents is too large in TREC test collections and it is similar to practical RF. The asymptotic query is modified as follows:

$$q_\infty = \alpha q + \frac{\beta}{card(R)} \sum_{d' \in R} d' - \frac{\gamma}{card(V_j)} \sum_{d' \in V} d' \tag{3.16}$$

An enhanced greedy term selection algorithm, greedy weighted term selection algorithm, is used to select the query terms from relevant documents and/or non-relevant documents. The ultimate aim is to select good positive terms that can retrieve exactly relevant documents and good negative terms that can suppress the highly ranked but non-relevant documents. The detail algorithm is shown in Figure 3.9. Initially, the algorithm starts with no good terms but with a set R of relevant documents for a particular topic, a subset V_j of top j retrieved and non-relevant documents for the user query, the desired number n_P of positive terms and n_N of negative terms in the output query Q_{∞,n_P,n_N}, and an initial set q of the terms in the user query. In step 2, all the terms in the relevant documents and non-relevant documents are added to the P' and N', respectively. In steps 3 and 4, any terms in P' and N' are stemmed and then added to positive set P and negative set N, respectively, if they are not stop words and not numerals, and they must occur in more than one document in the collection C. In step 5, the weight of each term t in positive set P and each term t in negative set N is calculated according to the term ranking function $\omega(t)$. Next, the terms are ranked by these weights in positive ranked term list L_P and negative ranked term list L_N. In steps 7 and 8, the positive query Q_{∞,n_P} and the negative query Q_{∞,n_N} are generated by using the top n_p terms and top n_N terms in the ranked term list L_P and L_N, respectively. This selection is based on the negative term selection scheme $\phi(L_P, L_N, n_P, n_N)$ in Table 3.10. Finally, use term weight function $\sigma_q(t)$ to calculate the weights of user query q, and use $\sigma_f(t)$ to calculate the weights of the terms in positive query Q_{∞,n_P} and negative query Q_{∞,n_N}. Then mix the output query with parameters α, β and γ and return.

Figure 3.10 shows various subsets of terms that can be selected to assign positive and negative weights. This figure is used to explain some terminologies in the negative term selection scheme. Let TC be all the terms in the document collection. Let TR be the set of terms in the relevant document for a particular topic. Let TV_j be the set of terms in the top j retrieved and non-relevant documents for the user query q. These two subsets of terms can be used to define three smaller subsets of terms that are useful for defining term selection schemes as follows. Let us define $A_j = TR - TV_j$. This set of terms appeared in the relevant documents but not in the non-relevant documents of the top j retrieved documents. These terms are like beacons of the relevant documents and they are weighted positively. Another set B_j of terms is the intersection of TR and TV_j. These terms appeared in both the relevant and non-relevant documents of the top j retrieved documents and these terms do not seem to have any discrimination ability. In general, these terms could

```
Method: Greedy Weighted Term Selection Algorithm
Input: set $R$, set $V_j$, set $q$, integer $n_P$, integer $n_N$, integer $\alpha$, integer $\beta$, integer $\gamma$
Output: set $Q_{\infty,n_P,n_N}$

  1. set $P \leftarrow \emptyset$, $N \leftarrow \emptyset$;
  2. set $P' \leftarrow \bigcup_{d \in R} d$, $N' \leftarrow \bigcup_{d \in V_j} d$;
  3. for each term $t \in P'$ do
      (a) if $t$ is a stop word then goto step 3;
      (b) $t' \leftarrow stem(t)$;
      (c) if $t' \in q$ then goto step 3;
      (d) if $t'$ is a numerical term then goto step 3;
      (e) if $(DF(C, t') < 2)$ then goto step 3;
      (f) $P \leftarrow \{t'\} \cup P$;
  4. for each term $t \in N'$ do
      (a) if $t$ is a stop word then goto step 4;
      (b) $t' \leftarrow stem(t)$;
      (c) if $t' \in q$ then goto step 4;
      (d) if $t'$ is a numerical term then goto step 4;
      (e) if $(DF(C, t') < 2)$ then goto step 4;
      (f) $N \leftarrow \{t'\} \cup N$;
  5. for each term $t$ in $P$ and each term $t$ in $N$, calculate the weight by using term ranking
     function $\omega(t)$;
  6. rank all the terms in $P$ according to term weights in $L_P$ and rank all the terms in $N$
     according to term weights in $L_N$;
  7. select top $n_P$ terms from ranked term list $L_P$ to formulate the positive query $Q_{\infty,n_P}$
     by using negative term selection scheme $\phi(L_P, L_N, n_P, n_N)$
  8. select top $n_N$ terms from ranked term list $L_N$ to formulate the negative query $Q_{\infty,n_N}$
     by using negative term selection scheme $\phi(L_P, L_N, n_P, n_N)$
  9. $Q_{\infty,n_P,n_N} \leftarrow \alpha \sum_{t \in q} (\sigma_q(t) \times t) + \beta \sum_{t \in Q_{\infty,n_P}} (\sigma_f(t) \times t) - \gamma \sum_{t \in Q_{\infty,n_N}} (\sigma_f(t) \times t)$
```

Figure 3.9: Greedy weighted term selection algorithm for formulating weighted asymptotic query

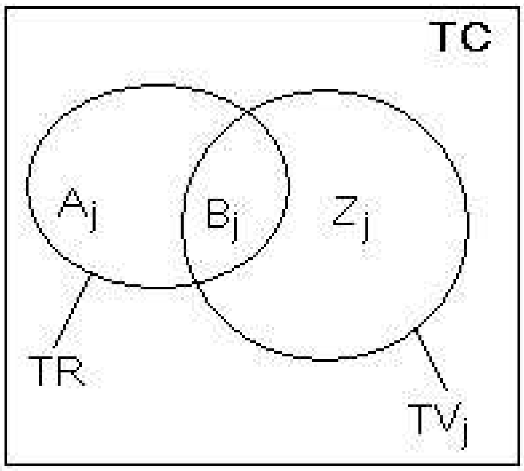


Figure 3.10: The different subsets of terms extracted from the relevant documents R and the top j retrieved and non-relevant document V_j

be assigned a zero weight. The set Z_j of terms is defined as $TV_j - TR$. These terms only appeared in the retrieved and non-relevant documents. Therefore, these terms should be strongly negatively weighted.

Six negative term selection schemes $\phi(.)$ can be defined using different combinations of the subsets of terms, A_j, B_j and Z_j. Table 3.10 shows the combination of these subsets for different term selection schemes. First, positively weighted scheme (1) is the same as to the previous section for selecting positively weighted terms. This scheme is included for comparison. Moreover, true positive terms scheme (2) only picks terms that appear in the relevant documents but not in the non-relevant documents of the top j retrieved documents. Therefore, extracting more (negatively weighted) terms for V_j from the top j retrieved and non-relevant documents reduces the number of terms in A_j (i.e., there are less "true positive" terms). However, true positive and true negative scheme (3) does not explicitly assign negative weights to terms during retrieval. In scheme (3), only terms that can indicate clearly whether documents are relevant or non-relevant are used because these terms only occur in the relevant documents or they only occur in the non-relevant and retrieved documents but not both. Furthermore, all subsets scheme (4)

No.	Scheme Name	Positively Weighted Terms	Negatively Weighted Terms
1	Positively Weighted	$A_j \cup B_j$	B_j
2	True Positive Terms	A_j	$\varnothing$
3	Ture Positive and True Negative	A_j	Z_j
4	All Subsets	$A_j \cup B_j$	$B_j \cup Z_j$
5	Negatively Weighted	B_j	$B_j \cup Z_j$
6	True Negative Terms	$\varnothing$	Z_j

Table 3.10: Different negative term selection schemes $\phi(.)$ based on different subsets of terms appearing in R and V_j

uses all the subsets of terms. Furthermore, negatively weighted scheme (5) selects terms that appear in the top j retrieved and non-relevant documents. Finally, true negative terms scheme (6) only picks terms that appear in the top j retrieved and non-relevant documents. Terms that appear in any of the relevant documents will not appear in the query for this scheme.

Figure 3.11 shows the MAP of different negative term selection schemes against different value of γ. This query is composed by adding top 175 ranked positively weighted terms and top 175 ranked negatively weighted terms to title queries. These negative weighted terms are extracted from the top 100 retrieved and non-relevant documents which use title query as the initial user query. There are two reasons for choosing top 100 retrieved documents: (1) The average number of relevant documents for each topic in TREC-6, 7 and 8 is about 100, and (2) From Figures 3.5 and 3.6, we find that the performance increased sharply until top 100. Moreover, the marginally best term ranking function $CHI_3 \times NSumNTF$ and the better term weight normalization scheme, rank normalization scheme (4), with $\alpha = 0.1$ and $\beta = 0.9$ is used in this experiment.

There are many interesting findings in Figure 3.11. Firstly, scheme (2) does

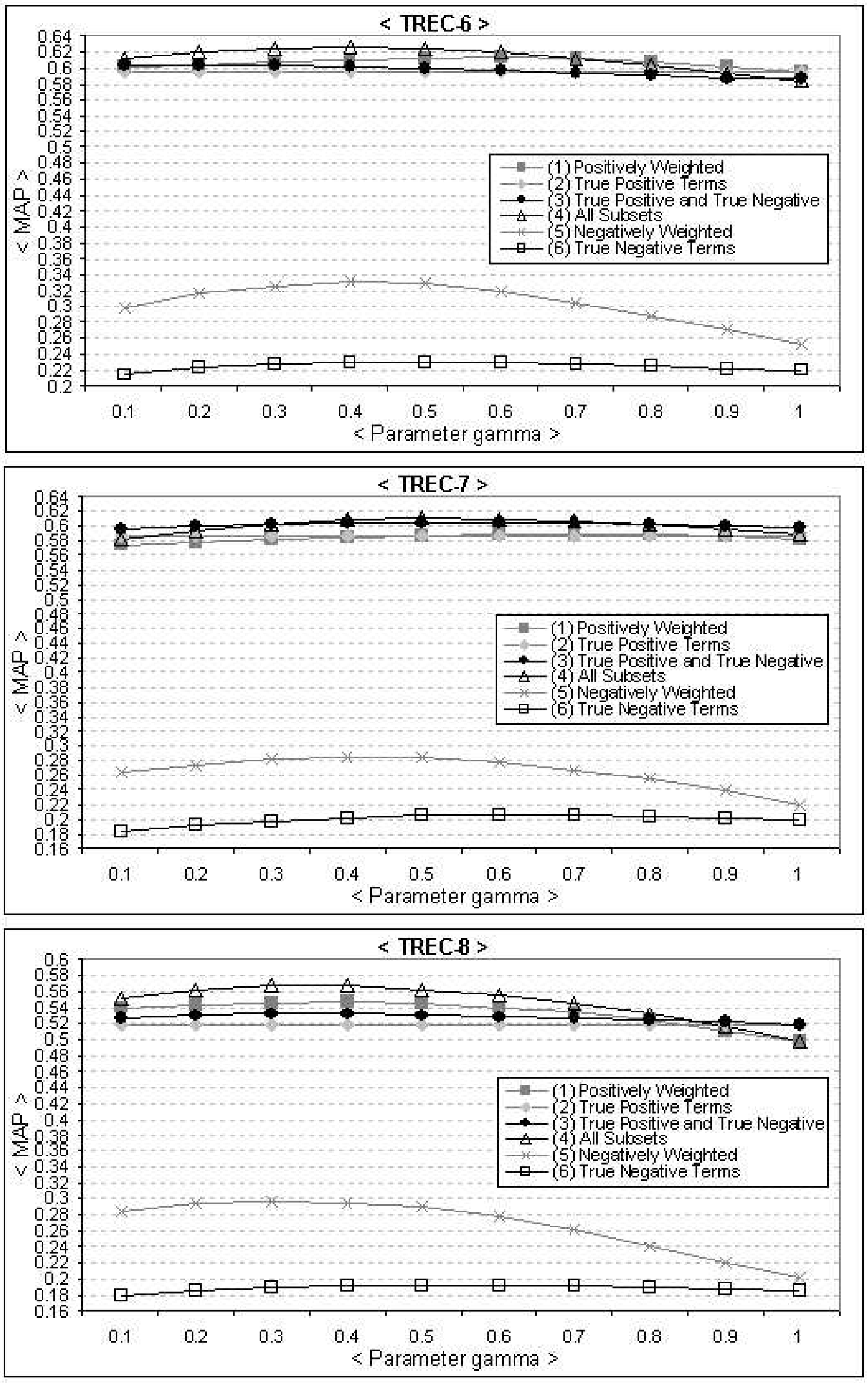


Figure 3.11: The MAP of negative term selection schemes $\phi(.)$ against the different parameter value γ.

not add any negatively weighted terms into the query, so that the γ does not have any effect on this scheme. Secondly, the MAP value of scheme (1) is larger than scheme (2), but after approaching a maximum, higher γ value has a negative impact on MAP for this scheme. Thirdly, the performance of scheme (3) is similar to scheme (2) because this scheme only uses terms with a clear signal that the terms should be weighted positively or negatively. Fourthly, the MAP of scheme (4) is the highest among the other schemes because this scheme includes all the subsets of terms (i.e., A_j, B_j and Z_j). Intuitively, this is expected since the impact of positively weighted terms does not necessarily decrease, and the number of terms lost in A_j is compensated for the increase in the number of terms in B_j. Fifthly, the best γ is 0.4 for scheme (4) so that the negative weights in the query will not overwhelm the positively weighted terms. Finally, schemes (5) and (6) are lower than the other schemes because these schemes only use negative impact.

In order to determine the suitable query size for negatively weighted terms, we picked the best two negative term selection schemes, the positively weighted scheme (1) with $\gamma = 0.6$, and the all subsets scheme (4) with $\gamma = 0.4$ and vary the query size from 0 to 300 to examine the effects. The experimental results are shown in Figure 3.12. Figure 3.12 shows that when more top ranked negative terms are added to the query using scheme (1) or scheme (4), the MAP value increases to a maximum and then begins taper off. The best MAP is achieved with about the top 250 negative terms by using scheme (4) with parameters $\alpha = 0.1$, $\beta = 0.9$ and $\gamma = 0.4$.

Table 3.11 shows the MAP value of negative term selection scheme (4) with the top 250 negative terms and with the best query size in each topic. The MAP value of the best query size is better. However, according to the results of the ANOVA test with 95% confidence, these improvements are not statistically significant. Hence, we concluded that the MAP values may not be improved by using various query sizes for different topics in negatively weighted terms.

Finally, from Table 3.12 we can see that, the improvement in MAP between

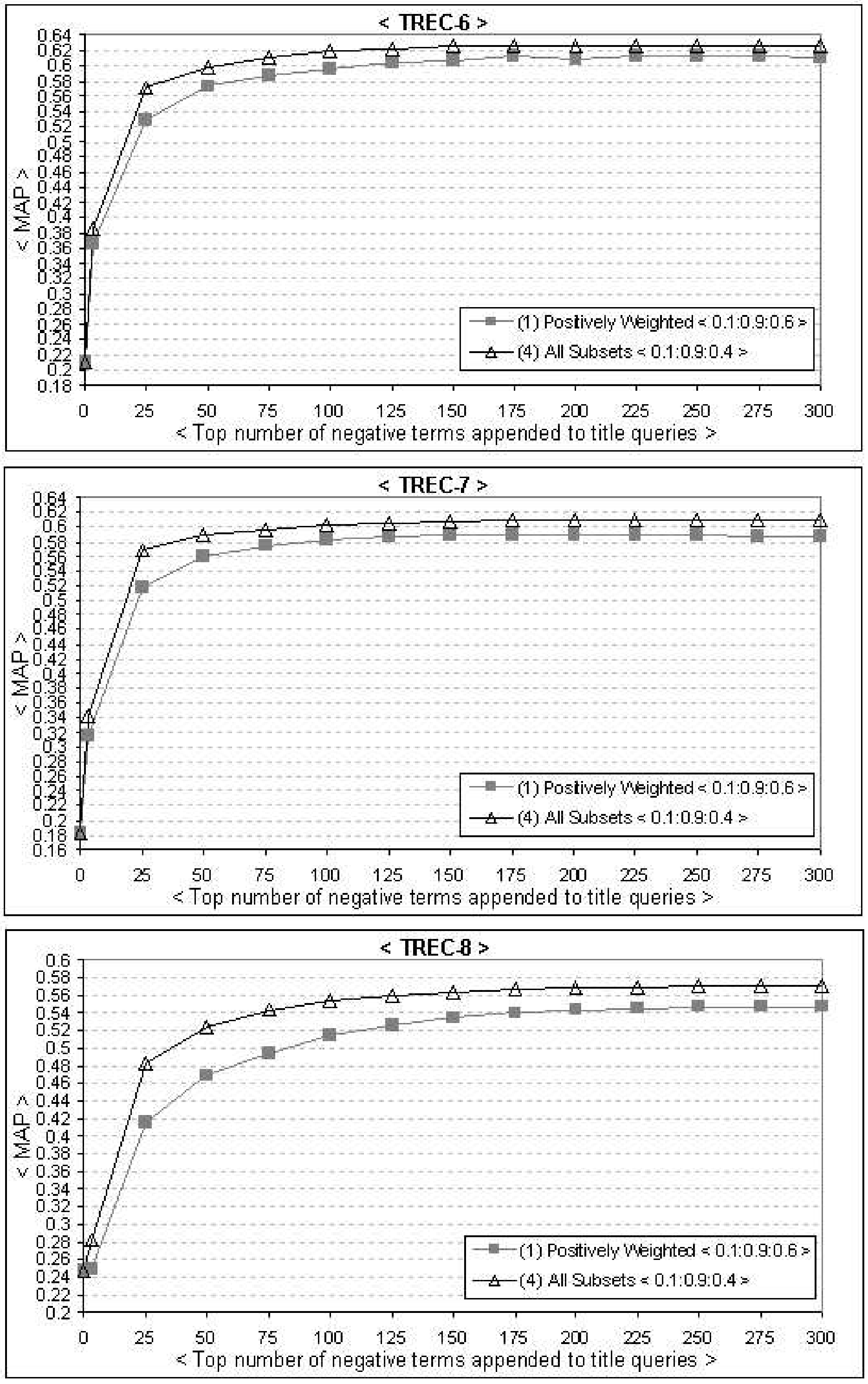


Figure 3.12: The MAP of IRF with negative term selection schemes against the different number of top n_N ranked negatively weighted terms.

	Top 250	Best size	BestSize>Top250 (95% conf.)
TREC-6	0.6250	0.6369	No
TREC-7	0.6101	0.6242	No
TREC-8	0.5711	0.5907	No

Table 3.11: The MAP of top 250 negative terms and the best query size in each topic using the best negative term selection scheme

	Unweighted (Un)	Positively Weighted (Pos)	Negatively Weighted (Neg)	Neg > Un (95% conf.)	Neg > Pos (95% conf.)
TREC-6	0.555	0.597	0.625	Yes	No
TREC-7	0.534	0.568	0.610	Yes	No
TREC-8	0.495	0.535	0.571	Yes	No

Table 3.12: The MAP of our IRF with unweighted terms, positively weighted terms and negatively weighted terms with fixed query size

the best MAP value achieved in this section (using both positively and negatively weighted terms, labeled 'Neg'), and the best MAP value achieved in Section 3.6 (using the top 175 ranked positively weighted terms, labeled 'Pos') is about 4%. This is not statistically significant with 95% confidence [VB02]. However, the improvement in MAP between the best MAP value achieved in this section and in Section 3.4 (using top the 100 ranked unweighted terms, labeled 'Un') is about 7%. This difference is statistically significant. Therefore, we conclude that negatively weighted terms have a significant improvement in MAP value.

3.8 Summary

The contribution of this chapter is to illustrate that Relevance Feedback in idealized situations can perform statistically significantly better than the corresponding practical Relevance Feedback which is constrained by the set of practical limitations.

We confirm this IRF and IPRF hypothesis in our experiments using three TREC test collections for English ad hoc retrieval. Moreover, this chapter reported a systematic study of the impact of different idealized RF situations that are summarized as assumptions in Table 3.1 on retrieval effectiveness. We conclude that:

- The user bias assumption (1) is valid and it has more impact if the number of dimensions is small.
- The experimental results in dimensionality reduction assumption (2) illustrates that the performance difference between the better term ranking functions $\omega(.)$ actually do not differ substantially using the log-odds ratio of Robertson and Karen Sparck Jones or the Chi-square or Kullback-Leibler divergence or the RSV by Robertson, provided either the average normalized term frequency or the average of the maximum of the normalized term frequencies are used.
- By exploring the asymptotic assumption (3), we find that using more top retrieved documents achieve better retrieval effectiveness independent of the initial query and the term ranking function $\omega(.)$. In addition, we realize that the IRF can be implemented by using a perfect classifier that identifies all relevant documents from the entire retrieval list of long query (i.e., TDN query) without errors. If we use title query rather than long query, the optimal retrieval effectiveness may not be achieved. It appears that the information in title query is not enough for optimal retrieval.
- By exploring the unweighted assumption (4), we conclude that if query terms are assigned weights, then better retrieval effectiveness can be achieved. The best term weights scheme $\sigma(.)$ is the rank normalization scheme (4), which assigns term weights based on the rank of the individual term.
- By exploring the non-negative weight assumption (5), we realize that if query terms are drawn from non-relevant documents and are assigned negative weights,

then significantly better retrieval effectiveness can be achieved. The best negative term selection scheme $\phi(.)$ is the all subsets scheme (4), which selects all the terms in relevant documents and in the top retrieved but non-relevant documents.

- The best MAP value of the various query sizes for different topics of any type of IRF: unweighted terms method, positively weighted terms method, and negatively weighted terms method, are not significantly better than that achieved using the fixed query size. This reveals that various query sizes for different topics may not be better. The best fixed query size for unweighted terms method is the top 100; for positively weighted terms method it is the top 175; and for negatively weighted terms method it is the top 175 positive terms and top 250 negative terms.

- The best MAP value of any type of IRF is significantly better than that achieved using the best interactive manual retrieval in the formal runs in TREC. It appears that a suitable choice of terms and a suitable choice of weights can substantially enhance the retrieval effectiveness of RF and PRF. The best MAP value for unweighted terms method is 62%, 59% and 56% for TREC-6,7 and 8, respectively; for positively weighted terms method it is 64%, 60% and 57% for TREC-6,7 and 8, respectively; and for negatively weighted terms method it is 64%, 62% and 59% for TREC-6, 7 and 8, respectively. In addition, for TREC-2005 robust track, the best MAP values of our IRF for unweighted terms method, positively weighted terms method and negatively weighted terms method is 0.5460, 0.6064 and 0.6297, respectively [WWL$^+$05]. These results are better than those known best in TREC-2005 robust track (i.e., $MAP = 0.332$).

The best MAP value of IRF is still below the optimal retrieval effectiveness (i.e., $MAP = 1.0$). Therefore, we will explore other methods in Section 4 to determine any better retrieval effectiveness that can be obtained.

Chapter 4

Combinatorial Optimization Search Strategy

This chapter applies combinatorial optimization search algorithms to find the near optimal query in order to determine if any better retrieval effectiveness can be reached. In Chapter 3, we looked at our IRF method that can in principle be used to find the near optimal query. However, the MAP value for TREC test collections is still below the optimal retrieval effectiveness. In this chapter, we propose to investigate the near optimal queries based on the view of the combinatorial optimization problem. From this perspective, the problem of finding an optimal query can be considered as the problem of searching for the best combination of terms out of all the possible combination of terms in all the relevant documents with binary term weights. We propose to apply local search algorithms to find the optimal queries within polynomial bounded computation times. We tested the ideas in our experiments using three TREC test collections for English ad hoc retrieval.

This chapter is organized as follows. Firstly, the background and literature review are given. Then, the optimization problem in finding the optimal query is formulated in Section 4.2. Moreover, three well-known local search methods: Hill Climbing, Best First Search and Simulated Annealing, are discussed and explored in Sections 4.3, 4.4 and 4.5, respectively. Section 4.6 introduces a novel search method, Combine Search. Definitions, algorithms and experimental results are presented. Finally, a summary of this chapter is given in Section 4.7.

4.1 Background and Literature Review

Combinatorial optimization problem is a kind of optimization problem in which the constraint functions are linear functions and the number of values in each optimization variable is finite. The aim of the combinatorial optimization problem is to find the best combination out of all possible combinations. There are some search methods that can find the global optimum solution of the combinatorial optimization problem in literature, such as the enumeration method and the branch and bound method. However, these methods are very time consuming especially when the search space is large. In order to control the search time, there are some faster search methods but these can only find the local optimum solution not the global optimum solution. Local search is an example of this method that promises to find the optimal solution within polynomial bounded computation times. Hill Climbing and Best First Search method, are examples of the local search method. Simulated Annealing and Genetic Algorithm are relatively slower local search methods but they are less likely to be trapped in local optima.

Local search methods in IR literature have been used widely. Some research has applied local search to discover term ranking functions [WYB88, FGP00, Ore02, LPGBA02, LPGBA03, FWX04, FGP05], whereas others have applied local search to automatically extract relations (e.g., hyponym relation) from documents [BJN00, Tro04]. Recently, local search has been applied to fuse multiple data sources [BBM02, ZCF^{+}05]. However, there are very few direct studies in IR literature that apply local search to find the optimal queries.

4.2 Modeling the problem of finding an optimal query

This section begins by describing our near optimal query searching problem. Then several functions used in the local search are defined in turn. Some of our IRF functions in Section 3 are altered in order to cater to the concepts of the local search.

4.2.1 Problem Description

The problem of finding an optimal query can be considered as the problem of searching for the best combination of terms out of all the combination of terms in the collection. However, when the collection is big, it is diffcult to implement such large searching space. In order to reduce the searching space, we simplify our problem to search for the best combination of terms out of all the combination of terms in all the relevant documents with binary term weights. That is, we assume that: (1) the optimal query term should appear in the relevant documents; and (2) the query term weights are not necessary for optimal retrieval. These assumptions are similar to assumptions (4) and (5) in Chapter 3.

The equation of finding an optimal query is defined in Equation 4.1. This equation is similar to the basic IRF equation (i.e., Equation 3.7) in Chapter 3, but it uses the residue of the relevant documents $\overline{U_m}$ in each state (i.e., iteration) m rather than the whole set of relevant documents R. This is because we are interested in investigating how many relevant documents have not been retrieved by the current query (i.e., current combination of terms):

$$q_m = q_{m-1} \cup \bigcup_{d' \in \overline{U_m}} d' \tag{4.1}$$

4.2.2 Objective Function

The objective of this problem is to find the optimal query q_o of F, such that:

$$q_o \equiv arg \min_{q \in F} cost(q) \tag{4.2}$$

```
Method: Cost function–$cost(.)$
Input: set $Q$, set $R$
Output: integer $c$, set $\overline{U}$

    1. select relevant documents from top $R$ retrieved documents of the query $Q$ into $U$;
    2. set the residue of relevant documents $\overline{U} \leftarrow (R - U)$;
    3. set the cost $c \leftarrow \frac{card(\overline{U})}{card(R)}$;
    4. return $c$ and $\overline{U}$.
```

Figure 4.1: Cost Function

where F is the feasible solution set of all the combinations of terms in the relevant documents of a particular topic. $cost(.)$ is the cost function to be defined in the next section.

4.2.3 Cost Function

A cost function is used to measure the effectiveness of the solutions. The detailed algorithm of our cost function $cost(.)$ is defined in Figure 4.1. The cost c is equal to $1-$R-prec of the query Q. The reason for choosing the R-prec measure but not the MAP measure is because calculating the R-prec measure is easier than the MAP measure. The set $\overline{U}$ in the algorithm is the relevant documents that are not present at the top R retrieved documents of the given query Q. The $card(\overline{U})$ and $card(R)$ are the number of documents in set $\overline{U}$ and in the relevant documents set R.

4.2.4 Neighborhood Function

A neighborhood function in local search is used to control the searching path (i.e., it decides the candidates in each search state). The neighborhood function for our problem can be considered as the term selection function in IR. We use our greedy term selection algorithm that was presented in Figure 3.1 on page 28 to be the neighborhood function $neighbor(.)$ except the terms are selected from the residue

```
Method: Neighborhood function–$neighbor(.)$
Input: set $\overline{U}$, integer $i$, set $q_{m-1}$
Output: set $q_{mi}$

  1. $P \leftarrow \emptyset$;
  2. $P' \leftarrow \bigcup_{d \in \overline{U}} d$;
  3. for each term $t \in P'$ do
       (a) if $t$ is a stop word then goto step 3;
       (b) $t' \leftarrow stem(t)$;
       (c) if $t' \in q_{m-1}$ then goto step 3;
       (d) if $t'$ is a numerical term then goto step 3;
       (e) if $(DF(C, t') < 2)$ then goto step 3;
       (f) $P \leftarrow \{t'\} \cup P$;
  4. for each term $t$ in $P$, calculate the weight by using term ranking function $\omega(t)$;
  5. rank all the terms in $P$ according to term weights in $L$;
  6. $q_{mi} \leftarrow (q_m \cup$ the $i^{th}$ term in the ordered term list $L)$;
  7. return $q_{mi}$.
```

Figure 4.2: Neighborhood function

set of relevant documents in each state (i.e., iteration) rather than from all the relevant documents. The detailed algorithm is shown in Figure 4.2. This neighborhood function returns a new query q_{mi} which is formulated by the given query q_{m-1} and the i^{th} neighbor of that given query.

4.2.5 Stopping Criteria

Stopping criteria are used to control the searching time (i.e., attempts) in local search. When a stopping criterion is satisfied, the searching process will be terminated and a local optimum will be returned. This local optimum is the optimal solution in the visited searching space. There are many methods to define the stopping criteria and the searching path in local search. The concepts in the Hill Climbing are the most simple and easy to understand in local search. Therefore, we begin by exploring the Hill Climbing search method in the next section.

4.3 Hill Climbing Search

Hill Climbing Search is a simple and efficient search method. The basic idea of the Hill Climbing Search is to head towards a state which is better than the current state. That is it only makes a move if the new query is better than the current query. This move is based on the path in the neighborhood function. Hill Climbing Search terminates when there are no neighbors (i.e., terms) of the current query that are better than the current query itself. Therefore, the chance of arriving at the goal state (i.e., optimal query) depends on the cost function and the neighborhood function.

The implementation of the Hill Climbing Search on finding the near optimal query is shown in Figure 4.3. Firstly, the search starts with the initial query q. Then the algorithm uses the cost function $cost(.)$, which is shown in Figure 4.1 to calculate the cost c_{local}. If the cost is equal to zero, the search terminates and the optimal query is returned. Then, in step 3, the query Q_{local} becomes the current query Q_{global} and the cost c_{local} becomes the current cost c_{global}. In step 3(d), a new query Q_{local} is formulated by the current query Q_{global} and a neighbor (i.e., a term) of current query. This neighbor is selected by the neighborhood function $neighbor(.)$ that is shown in Figure 4.2. Moreover, the new query Q_{local} is evaluated in step 3(e). Finally, if the cost of this new query is lower than the current query, then the new query achieves better R-prec value, so step 3 is repeated; otherwise, the search process is terminated and the current query Q_{global} is returned.

The Hill Climbing Search algorithm is evaluated using TREC-6 test collection and our VSM retrieval system. The term ranking function $\omega(t)$ of the neighborhood

```
Method: Hill Climbing Search algorithm
Input: set $R$, set $q$
Output: set $Q_{global}$

  1. $Q_{local} \leftarrow q$;
  2. $(c_{local}, \overline{U}_{local}) \leftarrow cost(Q_{local}, R, |R|_{doc})$;
  3. repeat
      (a) $Q_{global} \leftarrow Q_{local}$;
      (b) $\overline{U}_{global} \leftarrow \overline{U}_{local}$;
      (c) $c_{global} \leftarrow c_{local}$;
      (d) $Q_{local} \leftarrow neighbor(\overline{U}_{global}, Q_{global}, 1)$;
      (e) $(c_{local}, \overline{U}_{local}) \leftarrow cost(Q_{local}, R, |R|_{doc})$;
  4. until $c_{local} \geq c_{global}$;
  5. return $Q_{global}$.
```

Figure 4.3: Hill Climbing Search for finding the near optimal query

function is $CHI_3 \times NSumNTF$ (in Table 3.2, page 33), which is our best term ranking function in Chapter 3. In order to explore the impact of different starting points in the Hill Climbing Search, we compare the performance of the Hill Climbing Search starting with the title query and starting without any query (i.e., starting at zero). This is similar to examining the user bias assumption (1) in Chapter 3. The experimental results shows that the average R-prec when starting with the title query (=0.46) is slightly better than starting at zero (=0.44). However, these value are still far below the optimal effectiveness 1.0. In order to explore the potential causes of the poor performance of the Hill Climbing Search, we use three different views to observe this experimental results: (1) the R-prec values of the near optimal queries for each topics in the collection against the query size of that topic (i.e., Figure 4.4); (2) the R-prec against the different sizes of the relevant document set (i.e., Figure 4.5); and (3) the query size against the different sizes of the relevant document set (i.e., Figure 4.6). Moreover, there are two figures in any type of the

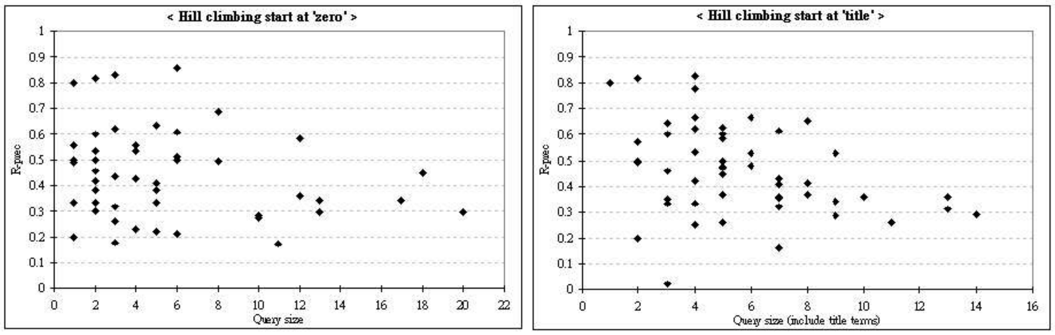


Figure 4.4: Comparing the R-prec with two different starting points against the different query sizes

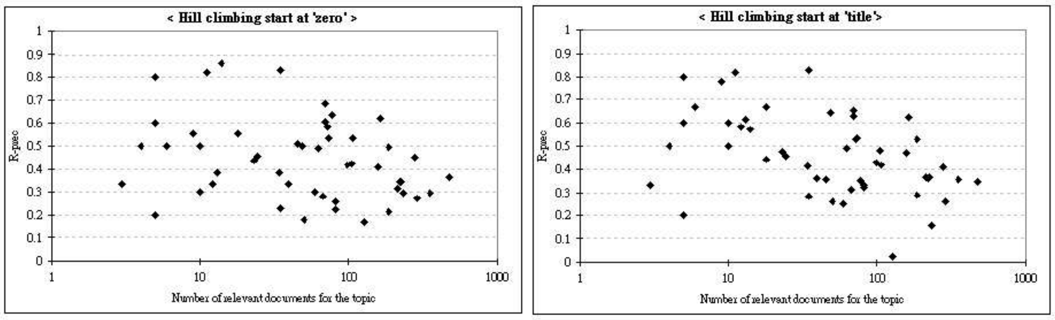


Figure 4.5: Comparing the R-prec with two different starting points against the different sizes of the relevant document set

views: the figure on the left side is the experimental results of the Hill Climbing Search starting at zero (labeled 'zero'), whereas the figure on the right side is starting with the title query (labeled 'title'). It should be noted that the query size in the experiment of starting at 'title' includes the number of terms in the title query except stopwords.

From Figures 4.4 and 4.5, we conclude that the performance of the Hill Climbing Search is independent of the sizes of the query and the sizes of the relevant document set in both types of the starting points. This performance is measured in the R-prec of the obtained query (i.e., the near optimal query). This reveals that

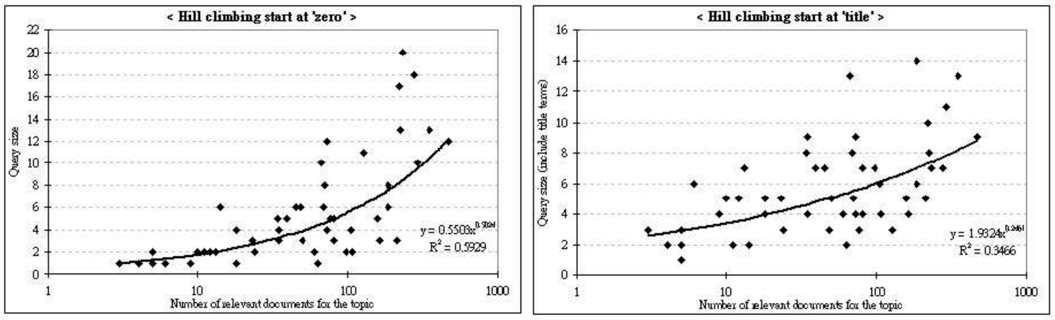


Figure 4.6: Comparing the query size with two different starting points against the different sizes of the relevant document set

more relevant documents in the collection may not represent poor retrieval effectiveness. Figure 4.6 illustrates that more relevant documents need more query terms to retrieve them. Moreover, Figure 4.4 shows that there are many optimal queries with sizes smaller than 2. This illustrates that the Hill Climbing Search algorithm stops very quickly for all types of starting points in many topics. The average searching time in the experiments are 6.38 and 4.24 attempts for starting at 'zero' and starting at 'title', respectively. One of the possible reasons for stopping too fast is that there are many local optima in the search space, so that it is very easy to be trapped in the local optimum.

The Hill Climbing method is a fast method to find the near optimal query because it does not attempt to exhaustively try every combination of terms in relevant documents. However, the average R-prec value it achieved is still far below the optimal retrieval effectiveness, and even below the retrieval effectiveness of any type of the IRF algorithms in Chapter 3. We believe that the main reason is trapped in the local optima in the search space. Therefore, in the next section, we explore another method, Best First Search, which is a slower method but may achieve better retrieval effectiveness. This method attempts more than one term to decide the optimal value in each state.

4.4 Best First Search

Best First Search is similar to Hill Climbing Search, but in Hill Climbing Search, once a move is chosen, the others are rejected and is never reconsidered. In contrast, in Best First Search the attempted terms are saved to enable revisits if an impasse occurs on the apparent best path. Basically, Best First Search is a combination of Depth First Search and Breadth First Search. Depth First Search is good because a solution can be found without computing all nodes. Breadth First is good because it does not get trapped in dead ends. The Best First Search allows us to switch between paths thus gaining the benefit of both approaches. If one of the terms chosen generates terms that are less promising, it is possible to choose another term at the same level and in effect the search changes from depth search to breadth search.

Figure 4.7 shows the algorithm of Best First Search on finding the near optimal query. This algorithm is similar to the Hill Climbing Search algorithm in Figure 4.3 except for step 3. In step 3, it attempts a number of neighbors n (i.e., breadth) of the current query Q_{global} rather than attempting only one neighbor. Then it chooses the best one to formulate the new query Q_{local}. Finally, if this new query achieves better R-prec value, then make it Q_{global} and repeat step 3; otherwise, return the query Q_{global} and terminate the search process.

In order to reduce the searching time, we only attempted 5 terms in each state (i.e., $n = 5$). The experiments are similar to the experiments in Section 4.3. We also used the TREC-6 test collection and two different starting points, 'zero' and 'title' to examine the impact of the starting point on retrieval effectiveness. The experimental results happened as expected, the retrieval effectiveness was improved but the searching time was increased. The average R-prec value of 'zero' and 'title' were 0.49 and 0.56, respectively, whereas, the average searching time of 'zero' and 'title' were 38.4 and 35.6 attempts, respectively. It is obvious that starting with the title query is significantly better than starting at zero. This observation is similar to

Method: Best First Search algorithm
Input: set R, set q, integer n
Output: set Q_{global}

```
1. $Q_{local} \leftarrow q$;
2. $(c_{local}, \overline{U}_{local}) \leftarrow cost(Q_{local}, R, |R|_{doc})$;
3. repeat
   (a) $Q_{global} \leftarrow Q_{local}$;
   (b) $\overline{U}_{global} \leftarrow \overline{U}_{local}$;
   (c) $c_{global} \leftarrow c_{local}$;
   (d) $i \leftarrow 1$;
   (e) $Q_{local} \leftarrow neighbor(\overline{U}_{global}, Q_{global}, i)$;
   (f) $(c_{local}, \overline{U}_{local}) \leftarrow cost(Q_{local}, R, |R|_{doc})$;
   (g) repeat
       i. $i \leftarrow (i+1)$;
       ii. $Q_i \leftarrow neighbor(\overline{U}_{global}, Q_{global}, i)$;
       iii. $(c_i, \overline{U}_i) \leftarrow cost(Q_i, R, |R|_{doc})$;
       iv. if $c_i < c_{local}$ then
           (I) $Q_{local} \leftarrow Q_i$;
           (II) $\overline{U}_{local} \leftarrow \overline{U}_i$;
           (III) $c_{local} \leftarrow c_i$;
   (h) until $i > n$;
4. until $c_{local} \geq c_{global}$;
5. return $Q_{global}$.
```

Figure 4.7: Best First Search for finding the near optimal query

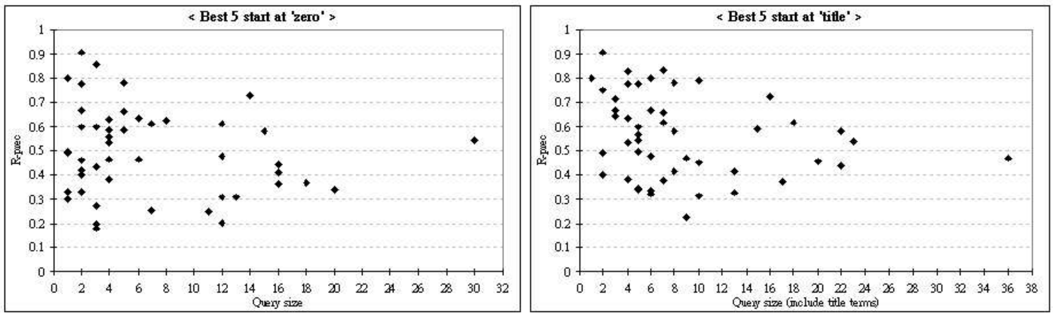


Figure 4.8: Comparing the R-prec with two different starting points against the different query sizes

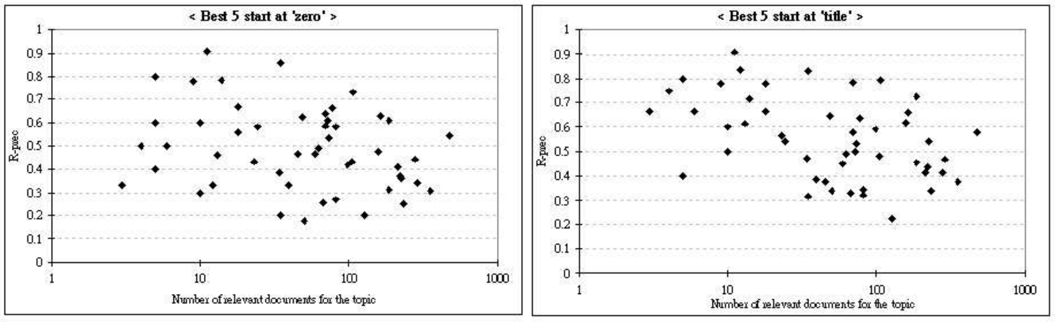


Figure 4.9: Comparing the R-prec with two different starting points against the different sizes of the relevant document set

the observation in the user bias assumption (1) in Chapter 3.

From Figures 4.8 and 4.9 we can conclude that the performance of the Best First Search is also independent of the sizes of the query and the sizes of the relevant document set in both types of starting points. Therefore, it is more likely to say that more relevant documents in the collection may not represent poor retrieval effectiveness. In addition, Figure 4.10 also illustrates that more relevant documents need more query terms to retrieve them. Furthermore, From Figure 4.4 we can see that there are one fifth of the optimal queries with a size smaller than 2. This implies that the Best First Search algorithm still stops very quickly on many topics.

Obviously, Best First Search algorithm achieves better retrieval effectiveness than Hill Climbing Search. However, the best retrieval effectiveness is still lower

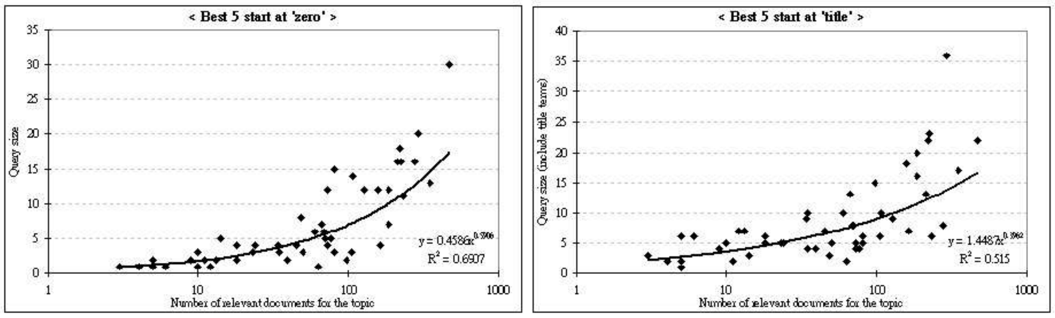


Figure 4.10: Comparing the query size with two different starting points against the different sizes of the relevant document set

than the IRF in Chapter 3. Therefore, we will continue our exploration of other local search methods that are less likely to be trapped in the local optima. Both Simulated Annealing and Genetic Algorithm are good for this kind of problem, but we will only examine Simulated Annealing Search method in this chapter because Genetic Algorithm is diffcult to implement when there are too many terms in the relevant document set.

4.5 Simulated Annealing Search

In the physical process of metallurgy, metals are heated to high temperatures and then cooled. The quality of the finished product depends on the cooling rate. A faster cooling rate leads to form large crystal structures and results in lower quality; this is equivalent to a local optimum. On the other hand, a suitable cooling rate can leads to a more uniform structure and results in better quality. This is equivalent to a global optimum. The heating and cooling processes are repeated until the final product is obtained. Simulated Annealing algorithm is designed based upon this phenomenon. The probability of making a large improvement move is lower than a small improvement move, and the probability of making large moves decreases with temperature. In addition, worse moves are also allowed but under some specific conditions. The acceptable probability p that the metal will make an improvement move (i.e., will jump to a higher energy level), is given by Equation 4.3, where k is Boltzmann's constant, ΔE is the change in the cost of the objective function, and T is the temperature:

$$p = \exp^{\frac{-\Delta E}{kT}} \tag{4.3}$$

The implementation of the Simulated Annealing Search on finding the near optimal query is shown in Figure 4.11. This algorithm is the refinement of Best First Search algorithm in Figure 4.7. In steps 3(i), 3(j) and 4, if the new query Q_{local} is not better then the current query Q_{global}, then we still accept the new query if the acceptable probability p (in Equation 4.3) is larger than our experimentally suggested threshold of 0.7, where ΔE is defined as the difference between the cost of Q_{global} and Q_{local}, and the constant k is set to 1. The temperature T is controlled by the annealing schedule in Equation 4.4, where T_j is the temperature for the j^{th} non-improved state:

$$\begin{cases} T_j = 1 & \text{for } \Delta E > 0 \\ T_j = T_{j-1} \times 0.9 & \text{for } \Delta E \leq 0 \end{cases} \tag{4.4}$$

In this experiment, we only explore the effects on starting with the 'title' query because the previous results illustrated that starting with the 'title' query is better than starting at 'zero'. Similarly, we use TREC-6 test collection and set $n = 5$ to get the best among five neighbors in each state. Unexpectedly, this experiment failed because the search process stops too slowly. Even if we increase the acceptable probability value to 0.8, the annealing process still takes a lot of time. It seems that the algorithm of the annealing schedule and the probability of jumping to the next energy level are not very suitable for finding the next optimal query. Therefore, we will introduce a novel search algorithm in the next section that can enhance the search performance.

Method: Simulated Annealing algorithm
Input: set R, set q, integer n
Output: set Q_{global}

```
1. $Q_{local} \leftarrow q$, temperature $T \leftarrow 1$;
2. $(c_{local}, \overline{U}_{local}) \leftarrow cost(Q_{local}, R, |R|_{doc})$;
3. repeat
    (a) $Q_{global} \leftarrow Q_{local}$;
    (b) $\overline{U}_{global} \leftarrow \overline{U}_{local}$;
    (c) $c_{global} \leftarrow c_{local}$;
    (d) $i \leftarrow 1$;
    (e) $Q_{local} \leftarrow neighbor(\overline{U}_{global}, Q_{global}, i)$;
    (f) $(c_{local}, \overline{U}_{local}) \leftarrow cost(Q_{local}, R, |R|_{doc})$;
    (g) repeat
        i. $i \leftarrow (i + 1)$;
        ii. $Q_i \leftarrow neighbor(\overline{U}_{global}, Q_{global}, i)$;
        iii. $(c_i, \overline{U}_i) \leftarrow cost(Q_i, R, |R|_{doc})$;
        iv. if $c_i < c_{local}$ then
            (I) $Q_{local} \leftarrow Q_i$;
            (II) $\overline{U}_{local} \leftarrow \overline{U}_i$;
            (III) $c_{local} \leftarrow c_i$;
    (h) until $i > n$;
    (i) if $c_{local} \geq c_{global}$ then
        i. probability $p \leftarrow \exp(\frac{-(c_{global} - c_{local})}{T})$;
        ii. $T \leftarrow (T \times 0.9)$;
    (j) else
        i. $p \leftarrow 1$;
        ii. $T \leftarrow 1$;
4. until $p \leq 0.7$;
5. return $Q_{global}$.
```

Figure 4.11: Simulated Annealing Search for finding the near optimal query

4.6 Combine Search

In this section a novel search method is proposed that is a combination of Best First Search and Simulated Annealing Search, Combine Search. The methodology and implementation is described in detail with experiments evaluated using TREC-6, 7 and 8 collections.

4.6.1 Methodology

According to the experimental results in previous sections, we realize that finding the near optimal query in a reasonable search time is not an easy task. It does not simply depend on the number of relevant documents in the collection or the query size, but it seems that there are some relations between them. We observe that when the number of the residue set of relevant documents gets smaller or the query size gets larger, the next improvement move is more difficult to get to. Therefore, we propose to refine the breadth (i.e., the number of neighbors of the current query to be attempted) of the Best First Search algorithm and the annealing schedule of the Simulated Annealing Search algorithm based on the query size and the cost in each state dynamically. The enhanced breadth definition is shown in Equation 4.5, and the enhanced annealing schedule is shown in Equations 4.6 to 4.8:

$$n = ceil(c_{global} \times \|Q_{global}\|_1) \tag{4.5}$$

$$\begin{cases} T = \log_{100}(\|Q_{global}\|_1) & \text{for } \Delta E \neq 0 \\ T = T \times c_{global} & \text{for } \Delta E = 0 \end{cases} \tag{4.6}$$

$$\begin{cases} p = 1 & \text{for } \Delta E > 0 \\ p = 0 & \text{for } \Delta E < 0 \\ p = T & \text{for } \Delta E = 0 \end{cases} \tag{4.7}$$

$$\Delta E = c_{global} - c_{local} \tag{4.8}$$

In detail, if the new query achieves better R-prec value than the current query (i.e., $\Delta E > 0$), then this new query is accepted as the current query. Moreover, if the R-prec of the new query and the current query are equal, then this new query is still accepted when the acceptable probability p is larger than 0.05. This acceptable probability is calculated by the log of the number of query terms in the current query $\|Q_{global}\|_1$. The reason for normalizing the value with $\log_{100}$ is because the best query size of the unweighted query in our IRF is about 100. Furthermore, if the new query achieves poor R-prec value than the current query, then it will attempt other neighbors. The number of neighbors n that will be attempted is dependent on the number of query terms in the current query and the current cost. However, if the number of query terms in the current query is too small (e.g., smaller than 3) or too large (e.g., larger than 100), the number of neighbors n will be too small or too large, respectively. In order to filter this outlier, it is set to 5 if the value is smaller than 5, and it is set to 30 if the value is larger than 30. Therefore, the largest number of attempts in one state is 30 and the smallest number of attempts is 5. The function $ceil(x)$ gives the smallest integer $\geq x$.

There are two novel methods in our Combine Search. First, it accepts more than one term in one state if these terms can achieve better R-prec value than using a single term. This method can be used to handle the term dependence problem, such as bi-gram terms. The detailed algorithm is shown in Figure 4.12. In this algorithm, the Breadth search route switches between single term and combined terms according to the cost. If the cost of the query Q_i and the new query Q_{local} in this state are the same, then the i^{th} neighbor (i.e., term) and the new query Q_{local} are combined to a query $Q_{combine}$. The cost $c_{combine}$ of the query $Q_{combine}$ is evaluated. If this cost $c_{combine}$ is lower than the cost c_{local} of the new query, then the combined query $Q_{combine}$ will become the new query Q_{local} in this state. Of course, if this cost is lower than the cost c_{global} of current query Q_{global}, then the Breadth search will stop and will continue Depth search. Otherwise, it will continue Breadth search.

```
• if $c_i < c_{local}$ then
    1. $Q_{local} \leftarrow Q_i, \overline{U}_{local} \leftarrow \overline{U}_i, c_{local} \leftarrow c_i$;
    2. $Q_{combine} \leftarrow Q_{local}$;
• else if $c_i = c_{local}$ then
    1. $Q_{combine} \leftarrow Q_{combine} \bigcup lastterm(Q_i)$;
    2. $(c_{combine}, \overline{U}_{combine}) \leftarrow cost(Q_{combine}, R, m)$;
    3. if $c_{combine} < c_{local}$ then set $Q_{local} \leftarrow Q_{combine}$, $\overline{U}_{local} \leftarrow \overline{U}_{combine}$,
       $c_{local} \leftarrow c_{combine}$;
```

Figure 4.12: Combined Term Method

It should be noted that this is a variation of the Breadth search because it will immediately stop when the lower cost (i.e., lower than c_{global}) is achieved rather than attempt after all the possible neighbors.

Second, it attempts multiple neighborhood function to decide the search path rather than only one neighborhood function. This is based on the idea that the search path can be sepearated into several stages, and different stages are suitable for different neighborhood functions. This idea is presented in Figure 4.13 with three different neighborhood functions. In detail, the Combine Search starts at the first stage $g1$ with first neighborhood function and with starting point at 'title' query. The first neighborhood function selects the terms (i.e., neighbors) from the top R retrieved and relevant documents based on $CHI_3 \times NSumNTF$ term ranking function. If the search is terminated but the goal state has not been arrived at (i.e., the R-prec is lower than 1.0), then the second stage $g2$ with second neighborhood function is applied and is started with the optimal query obtained by $g1$. The second neighborhood function selects the terms from the top R retrieved and relevant documents based on $W4$ term ranking function. Again, if the search is terminated but the goal state still has not been reached, then the third stage $g3$ with third neighborhood function is applied with the starting point as the optimal query obtained by

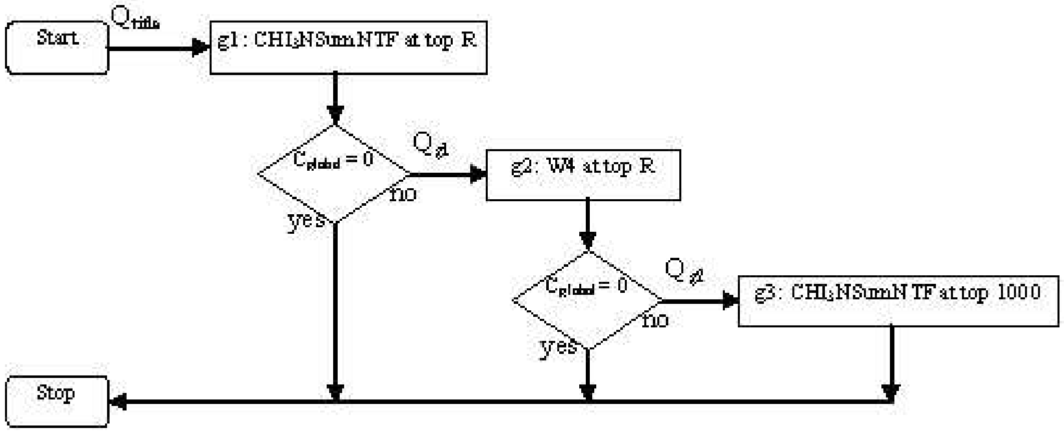


Figure 4.13: The overall search flow in Combine Search

$g2$. The third neighborhood function selects the terms from the top 1000 retrieved and relevant documents based on $CHI_3 \times NSumNTF$ term ranking function. Finally, when the search in stage $g3$ finishes, the Combine Search will finish no matter whether the goal state has been reached or not.

Figure 4.14 shows the search flow of one stage in Combine Search. It is obvious that our Combine Search is a combination of Best First Search and Simulated Annealing Search. Best First Search is good because a solution can be found without computing all the nodes and the search does not get trapped in dead ends. Simulated Annealing Search is good because a non improvement move is accepted. The Combine Search switches the search method based on the current cost and the current query size thus gaining the benefit of both approaches. The detailed algorithm of our Combine Search for finding the near optimal query is shown in Figure 4.15.

4.6.2 Experiments on Combine Search

The Combine Search algorithm is evaluated using TREC-6, 7 and 8 test collections and our VSM retrieval system. Table 4.1 shows the experimental results based on the different performance measures: (1) Retrieval Effectiveness, and (2) Search Performance.

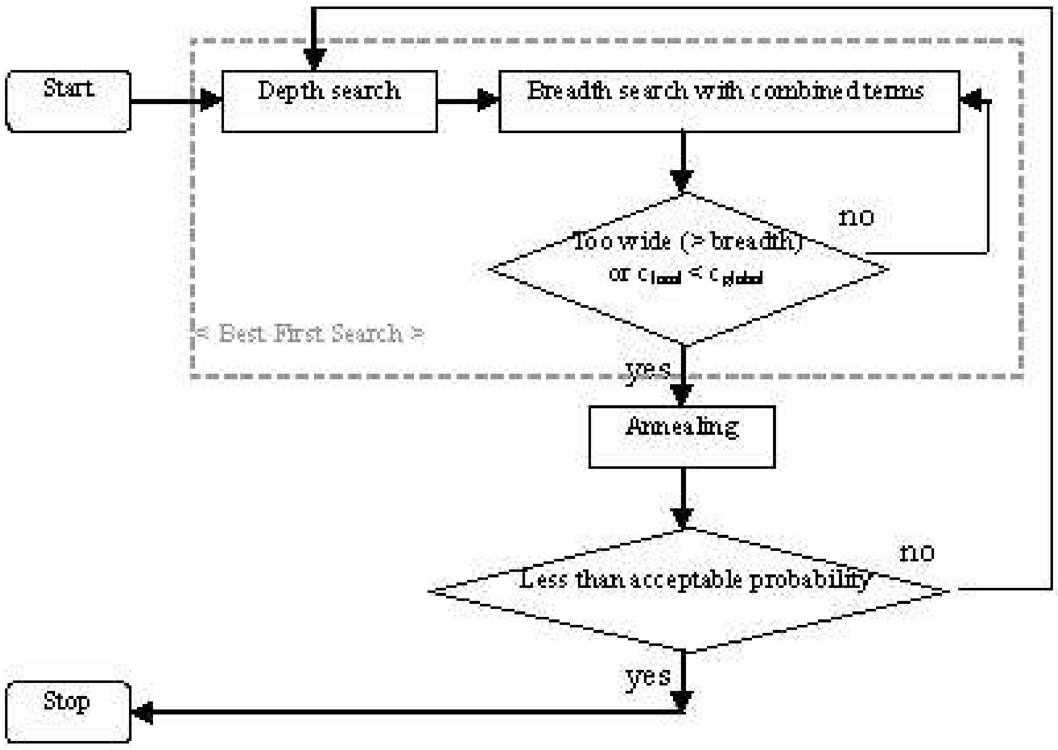

Start
Depth search
Breadth search with combined terms
Too wide (> breadth) or $c_{local} < c_{global}$
no
< Best-First Search >
yes
Annealing
Less than acceptable probability
no
yes
Stop

Figure 4.14: The search flow of one stage in Combine Search

	TREC-6	TREC-7	TREC-8	Avg
Retrieval Effectiveness				
R-prec	0.774	0.785	0.785	0.781
MAP	0.733	0.755	0.752	0.747
Queries with optimal effectiveness	6%	0%	0%	2%
Queries with MAP > 0.8	40%	38%	46%	41%
Queries with MAP < 0.5	4%	4%	4%	4%
Search Performance				
Avg search time	327	312	378	339
Avg query size	66	68	71	68
Terms obtained by breadth search	26%	27%	28%	25%
Terms obtained by simulated annealing	8%	8%	8%	8%
Terms obtained by combined term	11%	14%	14%	13%
Terms obtained by 2nd neighborhood func.	6%	11%	10%	9%
Terms obtained by 3th neighborhood func.	1%	1%	1%	1%

Table 4.1: Summary of the performance of the Combine Search method

Method: Combine Search algorithm
Input: set R, set q
Output: set Q_{global}

```
1. $Q_{local} \leftarrow q$, strategy $g \leftarrow 1$, the number of top documents $m \leftarrow |R|_{doc}$;
2. use $CHI_3NSumNTF$ as term ranking function $\omega(t)$;
3. repeat
    (a) $(c_{local}, \overline{U}_{local}) \leftarrow cost(Q_{local}, R, m)$;
    (b) repeat
        i. $Q_{global} \leftarrow Q_{local}$, $\overline{U}_{global} \leftarrow \overline{U}_{local}$, $c_{global} \leftarrow c_{local}$;
        ii. temperature $T \leftarrow log_{100}(|Q_{global}|_{term})$;
        iii. if $T > 1$ then set $T \leftarrow 1$;
        iv. the number of iteration $n \leftarrow ceil(c_{global} \times |Q_{global}|_{term})$;
        v. if $n < 5$ then set $n \leftarrow 5$, if $n > 30$ then set $n \leftarrow 30$;
        vi. $i \leftarrow 1$;
        vii. $Q_{local} \leftarrow neighbor(\overline{U}_{global}, Q_{global}, i)$;
        viii. $(c_{local}, \overline{U}_{local}) \leftarrow cost(Q_{local}, R, m)$;
        ix. $Q_{combine} \leftarrow Q_{local}$;
        x. repeat
            (I) $i \leftarrow (i + 1)$;
            (II) $Q_i \leftarrow neighbor(\overline{U}_{global}, Q_{global}, i)$;
            (III) $(c_i, \overline{U}_i) \leftarrow cost(Q_i, R, m)$;
            (IV) if $c_i < c_{local}$ then
                • $Q_{local} \leftarrow Q_i$, $\overline{U}_{local} \leftarrow \overline{U}_i$, $c_{local} \leftarrow c_i$;
                • $Q_{combine} \leftarrow Q_{local}$;
            (V) else if $c_i = c_{local}$ then
                • $Q_{combine} \leftarrow Q_{combine} \bigcup lastterm(Q_i)$;
                • $(c_{combine}, \overline{U}_{combine}) \leftarrow cost(Q_{combine}, R, m)$;
                • if $c_{combine} < c_{local}$ then
                    $Q_{local} \leftarrow Q_{combine}$, $\overline{U}_{local} \leftarrow \overline{U}_{combine}$, $c_{local} \leftarrow c_{combine}$;
        xi. until $(i > n)$ or $(c_{local} < c_{global})$;
        xii. if $c_{local} < c_{global}$ then set probability $p \leftarrow 1$;
        xiii. else if $c_{local} > c_{global}$ then set $p \leftarrow 0$;
        xiv. else
            (I) $T \leftarrow (T \times c_{global})$;
            (II) $p \leftarrow T$;
    (c) until $p \leq 0.05$;
    (d) if $c_{global} > 0$ then $Q_{local} \leftarrow Q_{global}$;
        i. if $g = 1$ then $g \leftarrow 2$, $m \leftarrow |R|_{doc}$, $\omega(t) \leftarrow W4(t)$
        ii. else if $g = 2$ then $g \leftarrow 3$, $m \leftarrow 1000$, $\omega(t) \leftarrow CHI_3NSumNTF(t)$
4. until $(g > 3)$ or $(c_{global} = 0)$;
5. return $Q_{global}$.
```

Figure 4.15: Combine Search for finding the near optimal query

Retrieval effectiveness indexes are used to measure the retrieval performance of the queries. From Table 4.1 we can see that the average R-prec and MAP of the three test collections are 0.781 and 0.747, respectively. This is much higher than the performance of the IRF in Chapter 3. Although there are only 2% real optimal queries (i.e., R-prec=1.0 and MAP=1.0), 41% of the queries have MAP larger than 0.8, and only 4% of the queries have MAP smaller than 0.5. This illustrates that a suitable choice of terms can substantially enhance the retrieval effectiveness in existing IR systems. In addition, the R-prec and MAP of the TREC-2005 robust track test collection is 0.743 and 0.711, respectively. These results indicate that the Combine Search algorithm also can achieve good retrieval effectiveness for hard topics.

Search Performance indexes are used to measure the performance of the search algorithms. Table 4.1 shows that the average search time of the three test collections is 339. This value is measured by the number of attempts in the search process rather than in terms of seconds. It is obvious that our Combine Search is much faster than trying every combinations of unique terms in the relevant document set, which is near $2^{7412} \approx 1.7E + 2231$ attempts. Moreover, we can see that the average query size is about 68 terms, in which title terms are included. This is shorter than the queries obtained by the IRF in Chapter 3. This reveals that the query terms that are obtained by Combine Search have more discriminability than the query terms obtained by our IRF.

In order to investigate how to obtain these good query terms, we sepearate our Combine Search algorithm into 5 modules: (1) breadth search method; (2) simulated annealing method; (3) combined term method; (4) second neighborhood function; and (5) third neighborhood function. The number of query terms that are obtained by those modules are shown at the bottom of Table 4.1. First, on average, there are 25% query terms that are obtained by using the benefit of the breadth search method. It appears that the optimal query terms are not always the highest ranking terms in the relevant document set. This reveals that the term ranking function still has room to improve. Second, on average there are only 8% and 13% query terms which are obtained by using the benefit of the simulated annealing method and the combined term method, respectively. This illustrates that the contribution of the simulated annealing method is less than the combined term method. This reveals that there are some dependencies between terms in the collection. We should select those terms altogether; otherwise, the search will be trapped in the local optima (i.e, the retrieval effectiveness will be poor). Finally, it seems that applying more than one neighborhood functions method is not very useful. On average, there are 9% and 1% query terms that are obtained by using the benefit of the second neighborhood function and third neighborhood function, respectively. It appears that using two different neighborhood functions is enough.

Figure 4.16 shows clearly the R-prec values of the near optimal queries for each topics in the three collections against the size of the query (on the left hand side of the figure) and the relevant document set (on the right hand side of the figure) of that topic. It appears that more attempts may not achieve higher retrieval effectiveness. Most of the real optimal queries (i.e., R-prec=1.0) are short queries and come from the topic with a small number of relevant documents in the collection. This can be illustrated clearly in the left hand side of Figure 4.17. From the figure we can see that the size of the near optimal queries for the topics with small relevant document sets are shorter than those with large relevant document sets. One of the possible reasons is that more documents include more information, so more query terms are needed to retrieve them. We believe that this information can be measured by the number of unique terms in the documents. However, the figure tells us that we cannot simply use the number of unique terms in the documents to measure the variety of the information in the documents. The figures show that the query size is more related to the number of relevant documents than the number of unique terms.

There are many topics that still cannot reach optimal retrieval effectiveness by using our Combine Search method. This may be due to the fact that either the

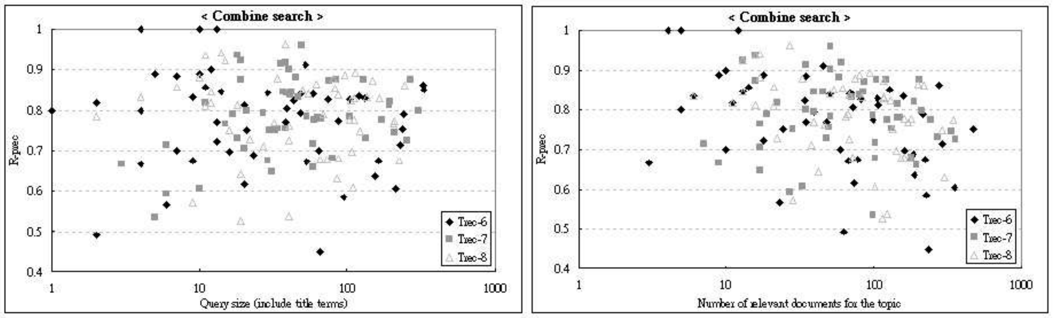


Figure 4.16: Comparing the R-prec against the query size and the number of relevant documents

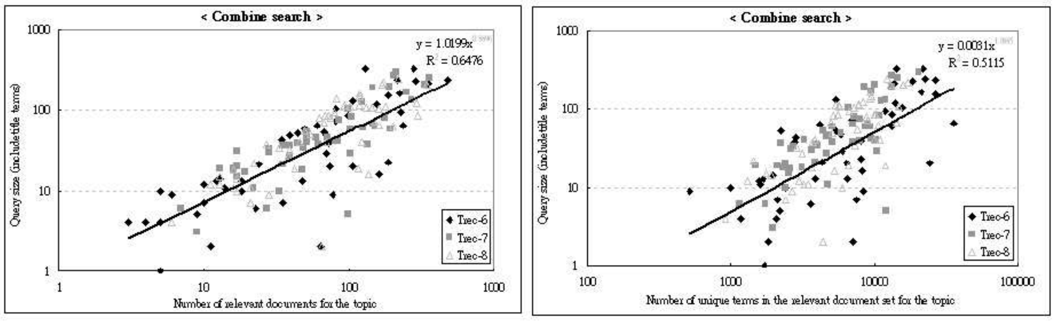


Figure 4.17: Comparing the query size against the number of relevant documents and the number of unique terms in the relevant document set

near optimal query should use the weights to give the different weighting to the query terms, or it should use the negative terms to discriminate the non-relevant documents from the collection. This is one of the possible future directions in our study.

4.7 Summary

The contributions of this chapter are to illustrate how to apply combinatorial optimization search algorithms to find the near optimal query and to verify that the main cause of the poor retrieval effectiveness in existing IR systems is the query factor. We believe that a suitable choice of terms can substantially enhance the retrieval effectiveness in existing IR systems.

Several kinds of local search methods are examined in turn. Hill Climbing Search, Best First Search and Simulated Annealing Search are implemented and evaluated with the TREC collections. However, the performance was worse than our Combine Search. The Combine Search method is a combination of Best First Search and Simulated Annealing Search. Best First Search is good because a solution can be found without computing all the nodes and the search does not get trapped in dead ends. Simulated Annealing Search is good because a non improvement move is accepted. The Combine Search switches the search method based on the current cost and the current query size, thus gaining the benefits of both approaches.

From the experimental results in this chapter, we conclude that:

- When the number of residual sets of relevant documents gets smaller or the query size gets longer, the query enhancement becomes more difficult. Based on the current query size and the current cost to define either the annealing schedule or the number of neighbors can enhance the performance effectively.
- The breadth search and the simulated annealing method can be used to escape

the local optima. It appears that the optimal query terms are not always the highest ranking terms in the relevant document set. This implies that there is still room to improve the term ranking function.

- Combined term method is an effective method to enhance the retrieval effectiveness. This method can be used to solve the term dependency problem.
- Using multiple neighborhood functions is slightly better than using single neighborhood function. This shows that the retrieval effectiveness impact of using different subsets of relevant documents as well as different term ranking functions is small.
- The optimal query size is related to the number of relevant documents for the topic but is not directly related to the number of unique terms in the relevant document set.

Chapter 5

Good Query Term Extraction

The experimental results in Chapters 2, 3 and 4 illustrates that a suitable choice of terms can substantially enhance the retrieval effectiveness in existing IR systems. Moreover, Chapter 3 shows that by using Relevance Feedback (RF) and Pseudo Relevance Feedback (PRF) in idealized situations, we can obtain about 60% MAP value for unweighted terms. Therefore, we believe that one of the possible research directions for improving the retrieval effectiveness in existing IR systems is to enhance the query terms selection method in PRF. In this chapter, we propose a novel query term expansion method, Good Query Term Extractor or GQTE, to expand the user query using a 'Good' query term classifier with PRF information. The terms in the near optimal queries that were obtained in Chapters 3 and 4 are used for training and testing the classifier with C4.5 classification method [Qui93]. Our GQTE is evaluated using the three TREC test collections for English ad hoc retrieval.

This chapter is organized as follows. It begins by introducing the C4.5 classification method, the divergence measure and the cross-validation procedure in Section 5.1. Then our GQTE is proposed in Section 5.2. The designs of the classification modules and the query generation module are described. Moreover, a discussion on the characteristics of the terms in the near optimal queries is presented in Section 5.3. The features which are used in GQTE are determined in this section. Furthermore, the implementation and practice in GQTE is described in Section 5.4. Our machine learning procedure, cross training, is introduced and

10-fold cross-validation is applied. Several methods for generating the new queries are also examined in this section. In addition, the evaluation based on estimation accuracy and retrieval effectiveness are reported. Finally, a summary of this chapter is given in Section 5.5.

5.1 Background

This section begins by introducing the C4.5 classification technique that is used for the classification module in GQTE. Then, a brief introduction on the divergence measure is given. The divergence is used for selecting the features with good discriminatory capabilities. Finally, a commonly used method for estimating the accuracy of a classifier, k-fold cross-validation, is mentioned.

5.1.1 C4.5 classification method

There are several kinds of methods for classification, such as the Nearest Neighbor rule [CH67], the Perceptron algorithm [Ros58], Support Vector Machines [HDO$^+$98] and Decision Tree [Qui93]. Some research [DHB90, SMT91, Qui94, Moo96] has compared these classification methods on a wide variety of problems. However, there seems to be no single type of classification method that is best for all the problems. Additionally, there do not seem to be any general conclusions that would enable one to say which classification method is best for which types of classification problems. In our study, we believe that a non-linear classifier is more suitable for our classification problem. Decision tree is one of a well-known non-linear classifier and is easy to derive an understandable rule set, so we use the decision tree classification method in our study.

A decision tree is a tree whose nodes are non-categorical attributes and whose arces are the attribute values. A leaf node of the tree specifies the expected value of the categorical attribute for the records described by the path from the root to that leaf. The most popular method for inducing decision trees is C4.5 [Qui93]. In C4.5, the decision tree at each node should be associated with the non-categorical attribute that is the most informative among the attributes not yet considered in the path from the root. This information is measured by the gain ratios. In addition, C4.5 accounts for missing values, interval attribute values and pruning of the decision trees. The pruning of the decision tree is done by replacing a whole subtree by a leaf node. The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf node.

5.1.2 Divergence Measure

The divergence is a well-known class separability measure [MG63]. It can be used to determine the discriminatory capabilities of the features. The divergence is a measure of statistical distance between the probability densities of the classes. For example, there are two classes A and B, if the distance between these two class densities $p_A(x)$ and $p_B(x)$ is greater for feature 1 than for feature 2, then we would like to use feature 1 for classification rather than feature 2. The divergence D is defined as:

$$D = \int_x [p_A(x) - p_B(x)] log_e \frac{p_A(x)}{p_B(x)} dx \tag{5.1}$$

where the integral is taken over the entire feature space. If the class densitites $p_A(x)$ and $p_B(x)$ are Gaussian densities with μ_A and μ_B and covariance matrices Σ_A and Σ_B then the divergence D can be revised as in Equation 5.2, where trM denotes the trace of matrix M, M^{-1} denotes the inverse of M, and M^T denotes the transpose of M. In our study, we assume that all of our class densities are Gaussian densities, so we use Equation 5.2 to calculate the divergence for all the features:

$$D = \frac{1}{2} tr[\Sigma_A - \Sigma_B][\Sigma_B^{-1} - \Sigma_A^{-1}] + \frac{1}{2} tr[\Sigma_A^{-1} + \Sigma_B^{-1}][\mu_A - \mu_B][\mu_A - \mu_B]^T \tag{5.2}$$

5.1.3 k-Fold Cross-Validation

In k-fold cross-validation, the dataset D is randomly split into k mutually exclusive subsets (i.e., the folds) D_1 to D_k of equal size. The induced classifier is trained and tested k times. In each time t, the induced classifier is trained on D_t and tested on the remaining subsets.

5.2 Design of the GQTE

In this section, a novel query term expansion method is proposed, Good Query Term Extractor or GQTE. It includes two modules: classification module and query generation module. The classification module is used to estimate the 'Good' query terms, whereas the query generation module is used to generate the estimated 'Good' query. The methodologies of each module are described in detail in the rest of this section.

5.2.1 Overall Structure

GQTE is a kind of query term expansion method based on PRF information. It reformulates the user query with the estimated 'Good' query terms that are selected from the top retrieved documents about the user query. This idea is proposed based on two assumptions: (1) the top retrieved documents about the user query are relevant to the user information's need (i.e., topic), which is similar to the assumption in PRF, and (2) the terms in the relevant documents of a particular topic can be classified into two classes: 'Good' and 'Normal'. 'Good' means the term is a good query term that has good retrieval capability, whereas 'Normal' means the term is not suitable to be a query term because its retrieval capability is poor. This assumption is likely to be valid if the 'Good' query terms have similar characteristics and have dissimilar characteristics to 'Normal' query terms for any topic in the collection, so that we can find a class boundary to sepearate them. We believe that this

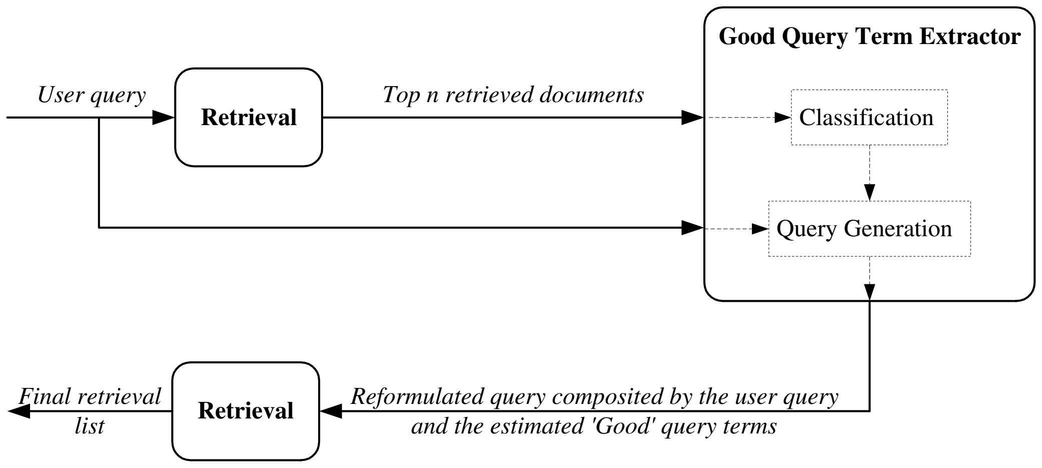

Good Query Term Extractor
User query
Retrieval
Top n retrieved documents
Classification
Query Generation
Final retrieval list
Retrieval
Reformulated query composited by the user query and the estimated 'Good' query terms

Figure 5.1: The mechanism of the Good Query Term Extractor

boundary is non-linear.

The overall structure of GQTE is shown in Figure 5.1. Firstly, the top n retrieved documents about the user query are obtained. Secondly, the terms in this set of documents are then sent to GQTE. The GQTE uses its classification module to estimate the possible class (i.e.,'Good' or 'Normal') of the given terms. The estimated 'Good' query terms are then sent to the query generation module to reformulate the user query using the defined weighting scheme and confidence level. Finally, the reformulated query is used to retrieve the documents again. This final retrieval list of documents is returned to the user.

5.2.2 Classification Module

The classification module in GQTE is used to estimate the 'Good' query terms. This classification module is a well trained classifier based on the C4.5 classification method. In detail, first, the classifier uses the pre-defined features to classify the incoming terms into two classes, 'Good' and 'Normal'. Then a list of 'Good' query terms is returned along with a confidence value. In this study, the classifier is trained by using the terms in our best near optimal queries as the 'Good' query terms, and using the remaining terms in the relevant document of that topic as the 'Normal' query terms. Our best near optimal query is the best among all the near optimal queries about a particular topic in Chapters 3 and 4. Moreover, in order to apply good discriminatory features into the classifier, we use a class separability measure, divergence, to select the relatively good features from a pile of possible features. The details are described in Section 5.3.

5.2.3 Query Generation Module

The query generation module in GQTE is used to generate an estimated 'Good' query about the user's information needs based on a user-defined confidence level. Firstly, the query generation module receives a set G of estimated 'Good' query terms from the classification module in GQTE. Secondly, the terms in G are filtered based on a user-defined confidence level. That is, if the confidence value of a term t is smaller than the confidence level, then the term t will be omitted. Therefore, the query size will be different for different topics because it is based on a confidence level rather than a fixed number. Finally, the reformulated query q_m is generated by using the Equation 5.3, where $\sigma_q(t)$ is a term weight functions for the user query q, $\sigma_f(t)$ is a term weight functions for the estimated 'Good' query terms, and α and β are parameters:

$$q_m \equiv \alpha \sum_{t \in q} \sigma_q(t)t + \beta \sum_{t \in G} \sigma_f(t)t \tag{5.3}$$

Four different term weight schemes are proposed to calculate the term weight functions of the user query $\sigma_q(t)$ and the estimated 'Good' query terms $\sigma_f(t)$. The details are shown in Table 5.1. The term weight schemes (1) and (2) are the best term weight schemes in Chapter 3. The term weight schemes (3) and (4) are newly proposed in this chapter. In particular, the unity weight scheme (1) is essentially without any weighting. The parameters α and β are equal to 1. The rank normalization scheme (2) is similar to the rank normalization scheme in Chapter 3 where the

		User query weight	'Good' query terms weight	Parameter	
No.	Scheme Name	$\sigma_q(t)$	$\sigma_f(t)$	α	β
1	Unity Weight	1	1	1	1
2	Rank Normalization	$\frac{1}{\lvert q\rvert_{term}}$	$\frac{\max_{t\in q_m}(rank(t)-rank(t)+1)}{\sum_{t\in q_m} rank(t)}$	0.1	0.9
3	Rank Normalization 2	$\frac{1}{\lvert q\rvert_{term}}$	$\frac{\max_{t\in q_m}(rank(t)-rank(t)+1)}{\sum_{t\in q_m} rank(t)}$	1	1
4	Confidence	1	conf(t)	1	1

Table 5.1: Term weight schemes for reformulating the user query with estimated 'Good' query terms

term weight in the user query q is normalized by the user query size; and the term weight in the estimated 'Good' query terms G is normalized by its rank $rank(t)$; and with the parameters α and β are also equal to 0.1 and 0.9, respectively. However, the rank in this chapter is ordered by the confidence value $conf(t)$ rather than the score of the term ranking function. If the terms have the same confidence value, the rank of these terms will be the same. Moreover, scheme (3) can be considered as a variation of scheme (2), where the parameters α and β are equal to 1. Finally, confidence scheme (4) uses the confidence of a term, $conf(t)$, to calculate the 'Good' query term weight directly. The user query weight are set to 1, and with the parameters α and β are also set to 1.

5.3 Features Selection

How to select features is an important issue in classification problems. In general, a good feature is a good discriminatory feature that can be used to separate the classes. In this study, we use the divergence measure to determine the discriminatory capability of the term, and to select the better discriminatory features. This divergence value is calculated using the Equation 5.2, in which the 'Good' class density $p_A(x)$ is calculated using the best near optimal query terms for 150 topics (i.e., all the topics in TREC-6, 7 and 8), and the 'Normal' class density $p_B(x)$ is calculated using the remaining terms in the relevant documents for that topic. In order to provide a balanced comparison, the number of 'Good' terms and 'Normal' terms for a topic is the same.

There are a lot of possible features for classifying the 'Good' and 'Normal' terms in the relevant documents. In the rest of this section, these possible features are introduced based on five major characteristic: (1) General Characteristic; (2) Location Characteristic; (3) Part-of-speech Characteristic; (4) Sentence Type Characteristic; and (5) Title Term Characteristic. The divergence of each feature is calculated, and then the final features using for the classification module in GQTE is determined by using this divergence value.

5.3.1 General Characteristic

General characteristic refers to the general term ranking functions, such as term frequency in the relevant document set and the document frequency in the collection. The detailed definitions of these kinds of features are shown in Table 5.2. Some of these features have been used for term ranking functions in Chapter 3. These features can be divided into three types: (1) 'relevant documents based' which includes Features 1 to 6, (2) 'collection based' which includes Features 7 to 11 and (3) 'both based' which includes Features 12 to 17. The features in Type (1) are mainly focused on the relevant document set, whereas the features in Type (2) are mainly focused on the collection. The features in Type (3) are focused on both the relevant document set and the collection. In general, Both Type (1) and Type (2) have two features calculated based on the document frequency, $DF(.)$ and $IDF(.)$. These two functions have been widely used in IR. Moreover, each type has four features: one that is based on the total term frequency (SumNTF), or the average term frequency (AvgNTF), or the maximum term frequency (MaxNTF), or the minimum term frequency (MinNTF) of a term in the documents. Furthermore, two more features are examined in Type (3), which is $CHI_3 * NSumNTF(t)$ (i.e., Feature 16), and the other is $W4*NMaxNTF(t)$ (i.e., Feature 17). These are the best two term ranking functions in Chapter 3, where with the detailed definitions shown in Item 7 of Table 3.2 and Equation 3.11 of Chapter 3, respectively.

The divergence value of each feature in General characteristic are shown in the last column of Table 5.2. It is obvious that Feature $CHI_3 * NSumNTF(t)$ and Feature $W4 * NMaxNTF(t)$ achieve the highest divergence value among all the features. Moreover, it appears that the features in Type (3) achieve higher divergence value than Types (1) and (2) on average. Furthermore, Features (1), (2), (7) and (8) perform poorly. This reveals that the document frequency is not a good feature for our classification problem. Finally, minimum term frequency (i.e., Features 6 and 11) is better than total, average and maximum term frequency in Type (1) and Type (2) but not in Type (3). In Type (3), total term frequency is better (i.e., Feature 12).

5.3.2 Location Characteristic

Location characteristic are the features related to the location of the query term in the relevant documents (i.e., 'Relevance Documents Based'). Several definitions are given. First, a paragraph is defined as a line of words that has a line break character at the last position and is preceded by a full stop, question mark or exclaimation mark character at the second to last position. Moreover, the first and last paragraph are defined as the line after the ⟨text⟩ tag and the line before the ⟨/text⟩ tag, respectively, and the body paragraphs are the lines between the first and last paragraph. Furthermore, a sentence is defined as a token of words with a full stop, question mark or exclaimation mark at the last position. However, if a token has a full stop at the last position and the number of characters in the token is smaller than 5, then this token is merged with the next token because it is likely to be an abbreviation

No.	Feature	Divergence
	(1) Relevant Documents Based	
1.	$DF(R,t) = card(r \in R, t \in r)$	0.528
2.	$IDF(R,t) = log\frac{card(R)+1}{DF(R,t)}$	0.739
3.	$SumNTF(R,t) = \sum_{d \in R} \frac{TF(d,t)}{\|d\|_1}$	6.109
4.	$AvgNTF(R,t) = \frac{SumNTF(R,t)}{card(R)}$	3.046
5.	$MaxNTF(R,t) = \max_{d \in R} \frac{TF(d,t)}{\|d\|_1}$	0.410
6.	$MinNTF(R,t) = \min_{d \in R} \frac{TF(d,t)}{\|d\|_1}$	**18.291**
	(2) Collection Based	
7.	$DF(C,t) = card(r \in C, t \in r)$	5.019
8.	$IDF(C,t) = log\frac{card(C)+1}{DF(C,t)}$	6.350
9.	$SumNTF(C,t) = \sum_{d \in C} \frac{TF(d,t)}{\|d\|_1}$	3.303
10.	$MaxNTF(C,t) = \max_{d \in C} \frac{TF(d,t)}{\|d\|_1}$	6.384
11.	$MinNTF(C,t) = \min_{d \in C} \frac{TF(d,t)}{\|d\|_1}$	**44.756**
	(3) Both Based	
12.	$NSumNTF(t) = \frac{SumNTF(R,t)}{SumNTF(C,t)}$	**29.342**
13.	$NAvgNTF(t) = \frac{AvgNTF(R,t)}{\frac{SumNTF(C,t)}{card(R)}}$	11.451
14.	$NMaxNTF(t) = \frac{MaxNTF(R,t)}{MaxNTF(C,t)}$	**15.600**
15.	$NMinNTF(t) = \frac{MinNTF(R,t)}{MinNTF(C,t)}$	3.697
16.	$CHI_3 * NSumNTF(t) = CHI_3(t) \times NSumNTF(t)$	**133.984**
17.	$W4 * NMaxNTF(t) = W4(t) \times NMaxNTF(t)$	**31.576**

Table 5.2: General Characteristics

word rather than a sentence. Finally, the first and last sentence are defined as the first and last token in a paragraph, and the body sentences are the tokens between the first and last sentence in a paragraph.

The features in the location characteristic can be divided into three types: (1) 'paragraph based' which is shown in Table 5.3 (i.e., Features 18 to 35); (2) 'paragraph and sentence based' which is shown in Table 5.4 (i.e., Features 36 to 62); and (3) 'paragraph and words based' which is shown in Table 5.5 (i.e., Features 63 to 71). In Type (1), the term frequency is counted when the query term appears in a particular paragraph, whereas in Type (2) the term frequency is counted when the query term appears in a particular sentence of a particular paragraph. In Type (3), the term frequency is counted when the query term appears in the top 30 words of a particular paragraph. The function $Location(p, s, d)$ shown in Tables 5.3, 5.4 and 5.5 refers to the term appearing in the sentence s of the paragraph p of the document d. For instance, $TF(Location(first, body, d), t)$ refers to the term frequency of the term t appearing in the body sentence of the first paragraph of the document d.

There are three types of inter-document normalization methods to normalize the term frequency in a document: one that is divided by the document size (NTF), one that is divided by the term frequency in the document (NTF2), and one that is divided by the number of words in the given location (NTF3), such as the first paragraph. Moreover, there are two types of intra-document normalization methods to calculate the term frequency in the relevant document set: one that is based on the total term frequency (Sum), and the other is based on the average term frequency (Avg) in all the relevant documents of the given topic. For instance, the feature, $SumNTFFirstWhole(R, t)$ (i.e., Feature 18 in Table 5.3), is calculated by three steps: step 1, the term frequency of the term t appearing in the first paragraph (i.e., 'FirstWhole') of a relevant document is calculated, step 2, the obtained term frequency is divided by the document size of that relevant document (i.e., 'NTF'), step 3, where step 1 is repeated until all the relevant documents are calculated. Step 4, the term frequency in each relevant document, are added together (i.e., Sum) and

No.	Feature	Divergence
18.	$SumNTFFirstWhole(R,t) = \sum_{d\in R} \frac{TF(Location(first,whole,d),t)}{\|d\|_1}$	14.443
19.	$AvgNTFFirstWhole(R,t) = \frac{SumNTFFirstWhole(R,t)}{card(R)}$	**<u>19.234</u>**
20.	$SumNTF2FirstWhole(R,t) = \sum_{d\in R} \frac{TF(Location(first,whole,d),t)}{TF(d,t)}$	3.212
21.	$AvgNTF2FirstWhole(R,t) = \frac{SumNTF2FirstWhole(R,t)}{card(R)}$	4.199
22.	$SumNTF3FirstWhole(R,t) = \sum_{d\in R} \frac{TF(Location(first,whole,d),t)}{\|Location(first,whole,d)\|_1}$	2.502
23.	$AvgNTF3FirstWhole(R,t) = \frac{SumNTF3FirstWhole(R,t)}{card(R)}$	1.841
24.	$SumNTFBodyWhole(R,t) = \sum_{d\in R} \frac{TF(Location(body,whole,d),t)}{\|d\|_1}$	**<u>15.672</u>**
25.	$AvgNTFBodyWhole(R,t) = \frac{SumNTFBodyWhole(R,t)}{card(R)}$	11.758
26.	$SumNTF2BodyWhole(R,t) = \sum_{d\in R} \frac{TF(Location(body,whole,d),t)}{TF(d,t)}$	3.331
27.	$AvgNTF2BodyWhole(R,t) = \frac{SumNTF2BodyWhole(R,t)}{card(R)}$	3.611
28.	$SumNTF3BodyWhole(R,t) = \sum_{d\in R} \frac{TF(Location(body,whole,d),t)}{\|Location(body,whole,d)\|_1}$	2.715
29.	$AvgNTF3BodyWhole(R,t) = \frac{SumNTF3BodyWhole(R,t)}{card(R)}$	1.813
30.	$SumNTFLastWhole(R,t) = \sum_{d\in R} \frac{TF(Location(last,whole,d),t)}{\|d\|_1}$	14.636
31.	$AvgNTFLastWhole(R,t) = \frac{SumNTFLastWhole(R,t)}{card(R)}$	**<u>18.698</u>**
32.	$SumNTF2LastWhole(R,t) = \sum_{d\in R} \frac{TF(Location(last,whole,d),t)}{TF(d,t)}$	3.464
33.	$AvgNTF2LastWhole(R,t) = \frac{SumNTF2LastWhole(R,t)}{card(R)}$	1.599
34.	$SumNTF3LastWhole(R,t) = \sum_{d\in R} \frac{TF(Location(last,whole,d),t)}{\|Location(last,whole,d)\|_1}$	2.703
35.	$AvgNTF3LastWhole(R,t) = \frac{SumNTF3LastWhole(R,t)}{card(R)}$	1.838

Table 5.3: Location Characteristic - (1) Paragraph Based

the final result is obtained. In addition, the equation $avg_{d\in R}$ in Tables 5.4 and 5.5 equal to $\frac{sum_{d\in R}}{card(R)}$.

From Tables 5.3, 5.4 and 5.5 we can see that the 'NTF' intra-document normalization method always achieves the highest divergence value among all the locations. On the other hand, the 'NTF3' normalization method always achieves the lowest divergence value. This implies that using 'NTF3' intra-document normalization method cannot generate good features for classification. Therefore 'NTF3' normalization method is not considered in the following experiments. Moreover, on average, the 'Avg' inter-document normalization method is better than the 'Sum' method. Therefore, we only use 'Avg' normalization method in the following experiments. Finally, it appears that the divergence values of the features in Type (1) (i.e.,

No.	Feature	Divergence
36.	$AvgNTFFirstFirst(R,t) = avg_{d\in R}\frac{TF(Location(first,first,d),t)}{\|d\|_1}$	3.691
37.	$AvgNTF2FirstFirst(R,t) = avg_{d\in R}\frac{TF(Location(first,first,d),t)}{TF(d,t)}$	7.324
38.	$AvgNTF3FirstFirst(R,t) = avg_{d\in R}\frac{TF(Location(first,first,d),t)}{\|Location(first,first,d)\|_1}$	0.247
39.	$AvgNTFFirstBody(R,t) = avg_{d\in R}\frac{TF(Location(first,body,d),t)}{\|d\|_1}$	4.755
40.	$AvgNTF2FirstBody(R,t) = avg_{d\in R}\frac{TF(Location(first,body,d),t)}{TF(d,t)}$	8.057
41.	$AvgNTF3FirstBody(R,t) = avg_{d\in R}\frac{TF(Location(first,body,d),t)}{\|Location(first,body,d)\|_1}$	0.302
42.	$AvgNTFFirstLast(R,t) = avg_{d\in R}\frac{TF(Location(first,Last,d),t)}{\|d\|_1}$	1.690
43.	$AvgNTF2FirstLast(R,t) = avg_{d\in R}\frac{TF(Location(first,Last,d),t)}{TF(d,t)}$	4.900
44.	$AvgNTF3FirstLast(R,t) = avg_{d\in R}\frac{TF(Location(first,Last,d),t)}{\|Location(first,Last,d)\|_1}$	0.045
45.	$AvgNTFBodyFirst(R,t) = avg_{d\in R}\frac{TF(Location(body,first,d),t)}{\|d\|_1}$	6.073
46.	$AvgNTF2BodyFirst(R,t) = avg_{d\in R}\frac{TF(Location(body,first,d),t)}{TF(d,t)}$	4.999
47.	$AvgNTF3BodyFirst(R,t) = avg_{d\in R}\frac{TF(Location(body,first,d),t)}{\|Location(body,first,d)\|_1}$	0.047
48.	$AvgNTFFBodyBody(R,t) = avg_{d\in R}\frac{TF(Location(body,body,d),t)}{\|d\|_1}$	3.096
49.	$AvgNTF2BodyBody(R,t) = avg_{d\in R}\frac{TF(Location(body,body,d),t)}{TF(d,t)}$	1.707
50.	$AvgNTF3BodyBody(R,t) = avg_{d\in R}\frac{TF(Location(body,body,d),t)}{\|Location(body,body,d)\|_1}$	0.245
51.	$AvgNTFBodyLast(R,t) = avg_{d\in R}\frac{TF(Location(body,Last,d),t)}{\|d\|_1}$	2.099
52.	$AvgNTF2BodyLast(R,t) = avg_{d\in R}\frac{TF(Location(body,Last,d),t)}{TF(d,t)}$	2.474
53.	$AvgNTF3BodyLast(R,t) = avg_{d\in R}\frac{TF(Location(body,Last,d),t)}{\|Location(body,Last,d)\|_1}$	0.170
54.	$AvgNTFLastFirst(R,t) = avg_{d\in R}\frac{TF(Location(Last,first,d),t)}{\|d\|_1}$	2.033
55.	$AvgNTF2LastFirst(R,t) = avg_{d\in R}\frac{TF(Location(Last,first,d),t)}{TF(d,t)}$	1.766
56.	$AvgNTF3LastFirst(R,t) = avg_{d\in R}\frac{TF(Location(Last,first,d),t)}{\|Location(Last,first,d)\|_1}$	0.086
57.	$AvgNTFLastBody(R,t) = avg_{d\in R}\frac{TF(Location(Last,body,d),t)}{\|d\|_1}$	1.516
58.	$AvgNTF2LastBody(R,t) = avg_{d\in R}\frac{TF(Location(Last,body,d),t)}{TF(d,t)}$	1.481
59.	$AvgNTF3LastBody(R,t) = avg_{d\in R}\frac{TF(Location(Last,body,d),t)}{\|Location(Last,body,d)\|_1}$	0.014
60.	$AvgNTFLastLast(R,t) = avg_{d\in R}\frac{TF(Location(Last,Last,d),t)}{\|d\|_1}$	2.012
61.	$AvgNTF2LastLast(R,t) = avg_{d\in R}\frac{TF(Location(Last,Last,d),t)}{TF(d,t)}$	1.564
62.	$AvgNTF3LastLast(R,t) = avg_{d\in R}\frac{TF(Location(Last,Last,d),t)}{\|Location(Last,Last,d)\|_1}$	0.252

Table 5.4: Location Characteristic - (2) Paragraph and Sentence Based

No.	Feature	Divergence
63.	$AvgNTFFirst30(R,t) = avg_{d \in R} \frac{TF(Location(first,30,d),t)}{\|d\|_1}$	4.586
64.	$AvgNTF2First30(R,t) = avg_{d \in R} \frac{TF(Location(first,30,d),t)}{TF(d,t)}$	1.426
65.	$AvgNTF3First30(R,t) = avg_{d \in R} \frac{TF(Location(first,30,d),t)}{\|Location(first,30,d)\|_1}$	0.072
66.	$AvgNTFBody30(R,t) = avg_{d \in R} \frac{TF(Location(body,30,d),t)}{\|d\|_1}$	6.248
67.	$AvgNTF2Body30(R,t) = avg_{d \in R} \frac{TF(Location(body,30,d),t)}{TF(d,t)}$	4.180
68.	$AvgNTF3Body30(R,t) = avg_{d \in R} \frac{TF(Location(body,30,d),t)}{\|Location(body,30,d)\|_1}$	0.437
69.	$AvgNTFLast30(R,t) = avg_{d \in R} \frac{TF(Location(last,30,d),t)}{\|d\|_1}$	5.452
70.	$AvgNTF2Last30(R,t) = avg_{d \in R} \frac{TF(Location(last,30,d),t)}{TF(d,t)}$	7.870
71.	$AvgNTF3Last30(R,t) = avg_{d \in R} \frac{TF(Location(last,30,d),t)}{\|Location(last,30,d)\|_1}$	0.269

Table 5.5: Location Characteristic - (3) Paragraph and Words Based

paragraph based) are better than Type (2) (i.e., paragraph and sentence based), and Type (3) (i.e., paragraph and words based). Therefore, we believe that the paragraph based features are good for our classification problem.

5.3.3 Part-Of-Speech Characteristic

Part-Of-Speech characteristic are the features related to the part-of-speech of the query term in the relevant documents (i.e., 'Relevance Documents Based'). We use Brill's tagger with Brown's corpus to determine the part-of-speech of the term. The part-of-speech tags can be divided into 5 classes: (a) JJ which includes adjectives, comparatives and superlatives; (b) NN which includes singular nouns and plural nouns; (c) NNP which includes proper noun both singular and plural; (d) RB which includes adverbs, compartives and superlatives; (e) VB which includes all the tenses of the verb (such as present tense, past tense and present participle tense).

The features in the part-of-speech (POS) characteristic can be divided into four types: (1) 'whole document based', which is shown in Table 5.6 (i.e., Features 72 to 81); (2) 'paragraph based', which is shown in Table 5.7 (i.e., Features 82 to 111); (3) 'paragraph and sentence based', which is shown in three tables: Table 5.8 (i.e.,

No.	Feature	Divergence
72.	$AvgNTFWholeJJ(R,t) = avg_{d \in R} \frac{TF(d,POS(JJ,t))}{\|d\|_1}$	0.479
73.	$AvgNTF2WholeJJ(R,t) = avg_{d \in R} \frac{TF(d,POS(JJ,t))}{TF(d,t)}$	3.001
74.	$AvgNTFWholeNN(R,t) = avg_{d \in R} \frac{TF(d,POS(NN,t))}{\|d\|_1}$	13.776
75.	$AvgNTF2WholeNN(R,t) = avg_{d \in R} \frac{TF(d,POS(NN,t))}{TF(d,t)}$	13.291
76.	$AvgNTFWholeNNP(R,t) = avg_{d \in R} \frac{TF(d,POS(NNP,t))}{\|d\|_1}$	10.838
77.	$AvgNTF2WholeNNP(R,t) = avg_{d \in R} \frac{TF(d,POS(NNP,t))}{TF(d,t)}$	10.622
78.	$AvgNTFWholeRB(R,t) = avg_{d \in R} \frac{TF(d,POS(RB,t))}{\|d\|_1}$	0.031
79.	$AvgNTF2WholeRB(R,t) = avg_{d \in R} \frac{TF(d,POS(RB,t))}{TF(d,t)}$	0.045
80.	$AvgNTFWholeVB(R,t) = avg_{d \in R} \frac{TF(d,POS(VB,t))}{\|d\|_1}$	1.081
81.	$AvgNTF2WholeVB(R,t) = avg_{d \in R} \frac{TF(d,POS(VB,t))}{TF(d,t)}$	1.188

Table 5.6: Part-Of-Speech Characteristic - (1) Whole Document Based

Features 112 to 129), Table 5.9 (i.e., Features 130 to 147) and Table 5.10 (i.e., Features 148 to 165); and (4) ‘paragraph and words based’, which is shown in Table 5.11 (i.e., Features 166 to 183). Type (1) only considers the part-of-speech characteristic, whereas Types (2), (3) and (4) consider both the part-of-speech and location characteristics. In addition, two better types of inter-document normalization method, ‘NTF’ and ‘NTF2’ are used and the ‘Avg’ intra-document normalization method is used.

Based only on consider the location characteristic, from Table 5.6 we can see that the noun and the proper noun (i.e., NN and NNP) calculate the term frequency to achieve the highest divergence value, whereas, the adverb and verb (i.e., RB and VB) achieve the lowest divergence value. If the location of the term is considered, Table 5.7 shows that the noun and proper noun method also achieve good divergence values, but the adverb and verb achieve poor divergence values. Therefore, the adverb and verb features are not considered in the following experiments. Moreover, from Tables 5.6 and 5.7 we can see that calculating the term frequency based on the paragraph location can achieve higher divergence value than the frequency based

No.	Feature	Divergence
82.	$AvgNTFFirstWholeJJ(R,t) = avg_{d \in R} \frac{TF(Location(first,whole,d),POS(JJ,t))}{\|d\|_1}$	9.890
83.	$AvgNTF2FirstWholeJJ(R,t) = avg_{d \in R} \frac{TF(Location(first,whole,d),POS(JJ,t))}{TF(d,t)}$	2.632
84.	$AvgNTFFirstWholeNN(R,t) = avg_{d \in R} \frac{TF(Location(first,whole,d),POS(NN,t))}{\|d\|_1}$	**48.437**
85.	$AvgNTF2FirstWholeNN(R,t) = avg_{d \in R} \frac{TF(Location(first,whole,d),POS(NN,t))}{TF(d,t)}$	9.417
86.	$AvgNTFFirstWholeNNP(R,t) = avg_{d \in R} \frac{TF(Location(first,whole,d),POS(NNP,t))}{\|d\|_1}$	**15.325**
87.	$AvgNTF2FirstWholeNNP(R,t) = avg_{d \in R} \frac{TF(Location(first,whole,d),POS(NNP,t))}{TF(d,t)}$	5.379
88.	$AvgNTFFirstWholeRB(R,t) = avg_{d \in R} \frac{TF(Location(first,whole,d),POS(RB,t))}{\|d\|_1}$	0.252
89.	$AvgNTF2FirstWholeRB(R,t) = avg_{d \in R} \frac{TF(Location(first,whole,d),POS(RB,t))}{TF(d,t)}$	0.037
90.	$AvgNTFFirstWholeVB(R,t) = avg_{d \in R} \frac{TF(Location(first,whole,d),POS(VB,t))}{\|d\|_1}$	3.556
91.	$AvgNTF2FirstWholeVB(R,t) = avg_{d \in R} \frac{TF(Location(first,whole,d),POS(VB,t))}{TF(d,t)}$	0.984
92.	$AvgNTFBodyWholeJJ(R,t) = avg_{d \in R} \frac{TF(Location(body,whole,d),POS(JJ,t))}{\|d\|_1}$	10.060
93.	$AvgNTF2BodyWholeJJ(R,t) = avg_{d \in R} \frac{TF(Location(body,whole,d),POS(JJ,t))}{TF(d,t)}$	1.931
94.	$AvgNTFBodyWholeNN(R,t) = avg_{d \in R} \frac{TF(Location(body,whole,d),POS(NN,t))}{\|d\|_1}$	**26.658**
95.	$AvgNTF2BodyWholeNN(R,t) = avg_{d \in R} \frac{TF(Location(body,whole,d),POS(NN,t))}{TF(d,t)}$	6.392
96.	$AvgNTFBodyWholeNNP(R,t) = avg_{d \in R} \frac{TF(Location(body,whole,d),POS(NNP,t))}{\|d\|_1}$	11.997
97.	$AvgNTF2BodyWholeNNP(R,t) = avg_{d \in R} \frac{TF(Location(body,whole,d),POS(NNP,t))}{TF(d,t)}$	5.828
98.	$AvgNTFBodyWholeRB(R,t) = avg_{d \in R} \frac{TF(Location(body,whole,d),POS(RB,t))}{\|d\|_1}$	0.014
99.	$AvgNTF2BodyWholeRB(R,t) = avg_{d \in R} \frac{TF(Location(body,whole,d),POS(RB,t))}{TF(d,t)}$	0.072
100.	$AvgNTFBodyWholeVB(R,t) = avg_{d \in R} \frac{TF(Location(body,whole,d),POS(VB,t))}{\|d\|_1}$	3.682
101.	$AvgNTF2BodyWholeVB(R,t) = avg_{d \in R} \frac{TF(Location(body,whole,d),POS(VB,t))}{TF(d,t)}$	0.823
102.	$AvgNTFLastWholeJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,whole,d),POS(JJ,t))}{\|d\|_1}$	14.183
103.	$AvgNTF2LastWholeJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,whole,d),POS(JJ,t))}{TF(d,t)}$	2.217
104.	$AvgNTFLastWholeNN(R,t) = avg_{d \in R} \frac{TF(Location(last,whole,d),POS(NN,t))}{\|d\|_1}$	**59.418**
105.	$AvgNTF2LastWholeNN(R,t) = avg_{d \in R} \frac{TF(Location(last,whole,d),POS(NN,t))}{TF(d,t)}$	8.979
106.	$AvgNTFLastWholeNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,whole,d),POS(NNP,t))}{\|d\|_1}$	**15.152**
107.	$AvgNTF2LastWholeNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,whole,d),POS(NNP,t))}{TF(d,t)}$	5.833
108.	$AvgNTFLastWholeRB(R,t) = avg_{d \in R} \frac{TF(Location(last,whole,d),POS(RB,t))}{\|d\|_1}$	0.040
109.	$AvgNTF2LastWholeRB(R,t) = avg_{d \in R} \frac{TF(Location(last,whole,d),POS(RB,t))}{TF(d,t)}$	0.027
110.	$AvgNTFLastWholeVB(R,t) = avg_{d \in R} \frac{TF(Location(last,whole,d),POS(VB,t))}{\|d\|_1}$	2.255
111.	$AvgNTF2LastWholeVB(R,t) = avg_{d \in R} \frac{TF(Location(last,whole,d),POS(VB,t))}{TF(d,t)}$	0.991

Table 5.7: Part-Of-Speech Characteristic - (2) Paragraph Based

No.	Feature	Divergence
112.	$AvgNTFFirstFirstJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,first,d),POS(JJ,t))}{\|d\|_1}$	0.073
113.	$AvgNTF2FirstFirstJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,first,d),POS(JJ,t))}{TF(d,t)}$	**34.054**
114.	$AvgNTFFirstFirstNN(R,t) = avg_{d\in R}\frac{TF(Location(first,first,d),POS(NN,t))}{\|d\|_1}$	6.101
115.	$AvgNTF2FirstFirstNN(R,t) = avg_{d\in R}\frac{TF(Location(first,first,d),POS(NN,t))}{TF(d,t)}$	**56.458**
116.	$AvgNTFFirstFirstNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,first,d),POS(NNP,t))}{\|d\|_1}$	5.947
117.	$AvgNTF2FirstFirstNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,first,d),POS(NNP,t))}{TF(d,t)}$	13.996
118.	$AvgNTFFirstBodyJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,body,d),POS(JJ,t))}{\|d\|_1}$	0.601
119.	$AvgNTF2FirstBodyJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,body,d),POS(JJ,t))}{TF(d,t)}$	2.862
120.	$AvgNTFFirstBodyNN(R,t) = avg_{d\in R}\frac{TF(Location(first,body,d),POS(NN,t))}{\|d\|_1}$	6.321
121.	$AvgNTF2FirstBodyNN(R,t) = avg_{d\in R}\frac{TF(Location(first,body,d),POS(NN,t))}{TF(d,t)}$	7.568
122.	$AvgNTFFirstBodyNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,body,d),POS(NNP,t))}{\|d\|_1}$	5.457
123.	$AvgNTF2FirstBodyNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,body,d),POS(NNP,t))}{TF(d,t)}$	11.007
124.	$AvgNTFFirstLastJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,last,d),POS(JJ,t))}{\|d\|_1}$	0.723
125.	$AvgNTF2FirstLastJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,last,d),POS(JJ,t))}{TF(d,t)}$	1.218
126.	$AvgNTFFirstLastNN(R,t) = avg_{d\in R}\frac{TF(Location(first,last,d),POS(NN,t))}{\|d\|_1}$	5.840
127.	$AvgNTF2FirstLastNN(R,t) = avg_{d\in R}\frac{TF(Location(first,last,d),POS(NN,t))}{TF(d,t)}$	7.161
128.	$AvgNTFFirstLastNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,last,d),POS(NNP,t))}{\|d\|_1}$	4.778
129.	$AvgNTF2FirstLastNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,last,d),POS(NNP,t))}{TF(d,t)}$	9.332

Table 5.8: Part-Of-Speech Characteristic - (3) Paragraph and Sentence Based <part 1>

on the whole document. Moreover, it appears that the divergence values of the features in Type (2) (i.e., paragraph based) are better than in Type (3) (i.e., paragraph and sentence based) and Type (4) (i.e., paragraph and words based). This is similar to the results in the location characteristic section. Therefore, we believe that the paragraph based features are good for our classification problem. We will only consider paragraph based features in the following experiments. Furthermore, comparing the features with the same type in location characteristic and part-of-speech characteristic (i.e., Table 5.7 vs. Table 5.3; Tables 5.8, 5.9 and 5.10 vs. Table 5.4; Table 5.11 vs. Table 5.5), we realize that using a combination of the part-of-speech and the location characteristics can achieve higher divergence value than using the location only. Therefore, we believe that using part-of-speech to calculate the term frequency can improve the classification performance.

No.	Feature	Divergence
130.	$AvgNTFBodyFirstJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,first,d),POS(JJ,t))}{\|d\|_1}$	0.122
131.	$AvgNTF2BodyFirstJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,first,d),POS(JJ,t))}{TF(d,t)}$	0.328
132.	$AvgNTFBodyFirstNN(R,t) = avg_{d\in R}\frac{TF(Location(body,first,d),POS(NN,t))}{\|d\|_1}$	5.807
133.	$AvgNTF2BodyFirstNN(R,t) = avg_{d\in R}\frac{TF(Location(body,first,d),POS(NN,t))}{TF(d,t)}$	6.016
134.	$AvgNTFBodyFirstNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,first,d),POS(NNP,t))}{\|d\|_1}$	7.465
135.	$AvgNTF2BodyFirstNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,first,d),POS(NNP,t))}{TF(d,t)}$	7.861
136.	$AvgNTFBodyBodyJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,body,d),POS(JJ,t))}{\|d\|_1}$	0.548
137.	$AvgNTF2BodyBodyJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,body,d),POS(JJ,t))}{TF(d,t)}$	1.471
138.	$AvgNTFBodyBodyNN(R,t) = avg_{d\in R}\frac{TF(Location(body,body,d),POS(NN,t))}{\|d\|_1}$	9.202
139.	$AvgNTF2BodyBodyNN(R,t) = avg_{d\in R}\frac{TF(Location(body,body,d),POS(NN,t))}{TF(d,t)}$	9.532
140.	$AvgNTFBodyBodyNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,body,d),POS(NNP,t))}{\|d\|_1}$	9.543
141.	$AvgNTF2BodyBodyNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,body,d),POS(NNP,t))}{TF(d,t)}$	10.049
142.	$AvgNTFBodyLastJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,last,,d),POS(JJ,t))}{\|d\|_1}$	0.635
143.	$AvgNTF2BodyLastJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,last,,d),POS(JJ,t))}{TF(d,t)}$	0.562
144.	$AvgNTFBodyLastNN(R,t) = avg_{d\in R}\frac{TF(Location(body,last,,d),POS(NN,t))}{\|d\|_1}$	2.271
145.	$AvgNTF2BodyLastNN(R,t) = avg_{d\in R}\frac{TF(Location(body,last,,d),POS(NN,t))}{TF(d,t)}$	**25.081**
146.	$AvgNTFBodyLastNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,last,,d),POS(NNP,t))}{\|d\|_1}$	4.800
147.	$AvgNTF2BodyLastNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,last,,d),POS(NNP,t))}{TF(d,t)}$	8.230

Table 5.9: Part-Of-Speech Characteristic - (3) Paragraph and Sentence Based <part 2>

5.3.4 Sentence Type Characteristic

Sentence type characteristic are the features related to the type of the sentence where the query term appears. The sentence type includes three types: (a) 'Stat' refers to the statement sentence; (2) 'Quest' refers to the question sentence; and (3) 'Exclaim' refers to the exclamation sentence.

The features in the sentence type characteristic can be divided into four types: (1) 'whole document based', which is shown in Table 5.12 (i.e., Features 184 to 189); (2) 'paragraph based', which is shown in Table 5.13 (i.e., Features 190 to 207); (3) 'whole document and POS based', which is shown in Table 5.14 (i.e., Features 208 to 225); (4) 'paragraph and POS based', which is shown in three tables: Table 5.15 (i.e., Features 226 to 243), Table 5.16 (i.e., Features 244 to 261) and Table 5.17 (i.e., Features 262 to 279). Type (1) only considers the sentence type characteristic. Type (2) combines the characteristics of the sentence type and location. Type (3)

No.	Feature	Divergence
148.	$AvgNTFLastFirstJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,first,d),POS(JJ,t))}{\|d\|_1}$	0.529
149.	$AvgNTF2LastFirstJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,first,d),POS(JJ,t))}{TF(d,t)}$	2.426
150.	$AvgNTFLastFirstNN(R,t) = avg_{d \in R} \frac{TF(Location(last,first,d),POS(NN,t))}{\|d\|_1}$	2.069
151.	$AvgNTF2LastFirstNN(R,t) = avg_{d \in R} \frac{TF(Location(last,first,d),POS(NN,t))}{TF(d,t)}$	1.408
152.	$AvgNTFLastFirstNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,first,d),POS(NNP,t))}{\|d\|_1}$	4.418
153.	$AvgNTF2LastFirstNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,first,d),POS(NNP,t))}{TF(d,t)}$	9.406
154.	$AvgNTFLastBodyJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,body,d),POS(JJ,t))}{\|d\|_1}$	0.653
155.	$AvgNTF2LastBodyJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,body,d),POS(JJ,t))}{TF(d,t)}$	1.910
156.	$AvgNTFLastBodyNN(R,t) = avg_{d \in R} \frac{TF(Location(last,body,d),POS(NN,t))}{\|d\|_1}$	2.943
157.	$AvgNTF2LastBodyNN(R,t) = avg_{d \in R} \frac{TF(Location(last,body,d),POS(NN,t))}{TF(d,t)}$	2.519
158.	$AvgNTFLastBodyNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,body,d),POS(NNP,t))}{\|d\|_1}$	11.711
159.	$AvgNTF2LastBodyNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,body,d),POS(NNP,t))}{TF(d,t)}$	9.604
160.	$AvgNTFLastLastJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,last,d),POS(JJ,t))}{\|d\|_1}$	2.548
161.	$AvgNTF2LastLastJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,last,d),POS(JJ,t))}{TF(d,t)}$	2.256
162.	$AvgNTFLastLastNN(R,t) = avg_{d \in R} \frac{TF(Location(last,last,d),POS(NN,t))}{\|d\|_1}$	2.236
163.	$AvgNTF2LastLastNN(R,t) = avg_{d \in R} \frac{TF(Location(last,last,d),POS(NN,t))}{TF(d,t)}$	3.396
164.	$AvgNTFLastLastNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,last,d),POS(NNP,t))}{\|d\|_1}$	4.392
165.	$AvgNTF2LastLastNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,last,d),POS(NNP,t))}{TF(d,t)}$	7.991

Table 5.10: Part-Of-Speech Characteristic - (3) Paragraph and Sentence Based <part 3>

No.	Feature	Divergence
166.	$AvgNTFFirst30JJ(R,t) = avg_{d \in R} \frac{TF(Location(first,30,d),POS(JJ,t))}{\|d\|_1}$	1.575
167.	$AvgNTF2First30JJ(R,t) = avg_{d \in R} \frac{TF(Location(first,30,d),POS(JJ,t))}{TF(d,t)}$	2.545
168.	$AvgNTFFirst30NN(R,t) = avg_{d \in R} \frac{TF(Location(first,30,d),POS(NN,t))}{\|d\|_1}$	4.789
169.	$AvgNTF2First30NN(R,t) = avg_{d \in R} \frac{TF(Location(first,30,d),POS(NN,t))}{TF(d,t)}$	7.568
170.	$AvgNTFFirst30NNP(R,t) = avg_{d \in R} \frac{TF(Location(first,30,d),POS(NNP,t))}{\|d\|_1}$	5.801
171.	$AvgNTF2First30NNP(R,t) = avg_{d \in R} \frac{TF(Location(first,30,d),POS(NNP,t))}{TF(d,t)}$	5.539
172.	$AvgNTFBody30JJ(R,t) = avg_{d \in R} \frac{TF(Location(body,30,d),POS(JJ,t))}{\|d\|_1}$	3.876
173.	$AvgNTF2Body30JJ(R,t) = avg_{d \in R} \frac{TF(Location(body,30,d),POS(JJ,t))}{TF(d,t)}$	1.226
174.	$AvgNTFBody30NN(R,t) = avg_{d \in R} \frac{TF(Location(body,30,d),POS(NN,t))}{\|d\|_1}$	6.553
175.	$AvgNTF2Body30NN(R,t) = avg_{d \in R} \frac{TF(Location(body,30,d),POS(NN,t))}{TF(d,t)}$	9.653
176.	$AvgNTFBody30NNP(R,t) = avg_{d \in R} \frac{TF(Location(body,30,d),POS(NNP,t))}{\|d\|_1}$	10.295
177.	$AvgNTF2Body30NNP(R,t) = avg_{d \in R} \frac{TF(Location(body,30,d),POS(NNP,t))}{TF(d,t)}$	4.718
178.	$AvgNTFLast30JJ(R,t) = avg_{d \in R} \frac{TF(Location(last,30,d),POS(JJ,t))}{\|d\|_1}$	0.572
179.	$AvgNTF2Last30JJ(R,t) = avg_{d \in R} \frac{TF(Location(last,30,d),POS(JJ,t))}{TF(d,t)}$	2.622
180.	$AvgNTFLast30NN(R,t) = avg_{d \in R} \frac{TF(Location(last,30,d),POS(NN,t))}{\|d\|_1}$	13.807
181.	$AvgNTF2Last30NN(R,t) = avg_{d \in R} \frac{TF(Location(last,30,d),POS(NN,t))}{TF(d,t)}$	9.395
182.	$AvgNTFLast30NNP(R,t) = avg_{d \in R} \frac{TF(Location(last,30,d),POS(NNP,t))}{\|d\|_1}$	2.575
183.	$AvgNTF2Last30NNP(R,t) = avg_{d \in R} \frac{TF(Location(last,30,d),POS(NNP,t))}{TF(d,t)}$	5.482

Table 5.11: Part-Of-Speech Characteristic - (4) Paragraph and Words based

No.	Feature	Divergence
184.	$AvgNTFWholeStat(R,t) = avg_{d \in R} \frac{TF(Location(whole,stat,d),t)}{\|d\|_1}$	8.964
185.	$AvgNTF2WholeStat(R,t) = avg_{d \in R} \frac{TF(Location(whole,stat,d),t)}{TF(d,t)}$	8.935
186.	$AvgNTFWholeQuest(R,t) = avg_{d \in R} \frac{TF(Location(whole,quest,d),POS(NN,t))}{\|d\|_1}$	**<u>18.971</u>**
187.	$AvgNTF2WholeQuest(R,t) = avg_{d \in R} \frac{TF(Location(whole,quest,d),POS(NN,t))}{TF(d,t)}$	4.808
188.	$AvgNTFWholeExclaim(R,t) = avg_{d \in R} \frac{TF(Location(whole,exclaim,d),POS(NNP,t))}{\|d\|_1}$	5.417
189.	$AvgNTF2WholeExclaim(R,t) = avg_{d \in R} \frac{TF(Location(whole,exclaim,d),POS(NNP,t))}{TF(d,t)}$	7.779

Table 5.12: Sentence Type Characteristic - (1) Whole Document Based

combines the charactertistics of the sentence type and part-of-speech. Type (4) combines the charatertistics of the sentence type, location and part-of-speech. In addition, 'NTF' and 'NTF2' inter-document normalization methods are used, and the 'Avg' intra-document normalization method is used.

For considering only the sentence type characteristic, from Table 5.12 we can see that based on the question sentence (i.e., Quest) to calculate the term frequency achieves the highest divergence value. Moreover, if the paragraph location of the term is considered, Table 5.13 shows that the divergence values are similar to that not considered the location. However, if the part-of-speech of the term is considered, Table 5.14 illustrates that the features based on the question sentence also achieve higher divergence value, especially for the noun and proper noun. Furthermore, if both the location and the part-of-speech are considered (features in Table 5.15, Table 5.16 and Table 5.17), it appears that in the first and last paragraph, the 'Stat' and 'Exclaim' features achieve better divergence values, whereas in the body paragraph, the 'Quest' features are better. In addition, it seems that the adjective features are weaker than the noun features and proper noun features. Finally, we believe that combing the paragraph location characteristic, the part-of-speech characteristic and the sentence type characteristic to calculate the term frequency properly can improve the classification performance.

No.	Feature	Divergence
190.	$AvgNTFFirstStat(R,t) = avg_{d \in R} \frac{TF(Location(first,stat,d),t)}{\|d\|_1}$	3.106
191.	$AvgNTF2FirstStat(R,t) = avg_{d \in R} \frac{TF(Location(first,stat,d),t)}{TF(d,t)}$	1.704
192.	$AvgNTFFirstQuest(R,t) = avg_{d \in R} \frac{TF(Location(first,quest,d),POS(NN,t))}{\|d\|_1}$	1.616
193.	$AvgNTF2FirstQuest(R,t) = avg_{d \in R} \frac{TF(Location(first,quest,d),POS(NN,t))}{TF(d,t)}$	7.746
194.	$AvgNTFFirstExclaim(R,t) = avg_{d \in R} \frac{TF(Location(first,exclaim,d),POS(NNP,t))}{\|d\|_1}$	2.696
195.	$AvgNTF2FirstExclaim(R,t) = avg_{d \in R} \frac{TF(Location(first,exclaim,d),POS(NNP,t))}{TF(d,t)}$	4.595
196.	$AvgNTFBodyStat(R,t) = avg_{d \in R} \frac{TF(Location(body,stat,d),t)}{\|d\|_1}$	1.642
197.	$AvgNTF2BodyStat(R,t) = avg_{d \in R} \frac{TF(Location(body,stat,d),t)}{TF(d,t)}$	6.933
198.	$AvgNTFBodyQuest(R,t) = avg_{d \in R} \frac{TF(Location(body,quest,d),POS(NN,t))}{\|d\|_1}$	0.329
199.	$AvgNTF2BodyQuest(R,t) = avg_{d \in R} \frac{TF(Location(body,quest,d),POS(NN,t))}{TF(d,t)}$	1.689
200.	$AvgNTFBodyExclaim(R,t) = avg_{d \in R} \frac{TF(Location(body,exclaim,d),POS(NNP,t))}{\|d\|_1}$	0.797
201.	$AvgNTF2BodyExclaim(R,t) = avg_{d \in R} \frac{TF(Location(body,exclaim,d),POS(NNP,t))}{TF(d,t)}$	6.336
202.	$AvgNTFLastStat(R,t) = avg_{d \in R} \frac{TF(Location(last,stat,d),t)}{\|d\|_1}$	12.011
203.	$AvgNTF2LastStat(R,t) = avg_{d \in R} \frac{TF(Location(last,stat,d),t)}{TF(d,t)}$	4.196
204.	$AvgNTFLastQuest(R,t) = avg_{d \in R} \frac{TF(Location(last,quest,d),POS(NN,t))}{\|d\|_1}$	0.076
205.	$AvgNTF2LastQuest(R,t) = avg_{d \in R} \frac{TF(Location(last,quest,d),POS(NN,t))}{TF(d,t)}$	6.780
206.	$AvgNTFLastExclaim(R,t) = avg_{d \in R} \frac{TF(Location(last,exclaim,d),POS(NNP,t))}{\|d\|_1}$	0.085
207.	$AvgNTF2LastExclaim(R,t) = avg_{d \in R} \frac{TF(Location(last,exclaim,d),POS(NNP,t))}{TF(d,t)}$	1.725

Table 5.13: Sentence Type Characteristic - (2) Paragraph Based

No.	Feature	Divergence
208.	$AvgNTFWholeStatJJ(R,t) = avg_{d \in R} \frac{TF(Location(whole,stat,d),POS(JJ,t))}{\|d\|_1}$	0.450
209.	$AvgNTF2WholeStatJJ(R,t) = avg_{d \in R} \frac{TF(Location(whole,stat,d),POS(JJ,t))}{TF(d,t)}$	2.941
210.	$AvgNTFWholeStatNN(R,t) = avg_{d \in R} \frac{TF(Location(whole,stat,d),POS(NN,t))}{\|d\|_1}$	13.591
211.	$AvgNTF2WholeStatNN(R,t) = avg_{d \in R} \frac{TF(Location(whole,stat,d),POS(NN,t))}{TF(d,t)}$	13.080
212.	$AvgNTFWholeStatNNP(R,t) = avg_{d \in R} \frac{TF(Location(whole,stat,d),POS(NNP,t))}{\|d\|_1}$	10.903
213.	$AvgNTF2WholeStatNNP(R,t) = avg_{d \in R} \frac{TF(Location(whole,stat,d),POS(NNP,t))}{TF(d,t)}$	10.426
214.	$AvgNTFWholeQuestJJ(R,t) = avg_{d \in R} \frac{TF(Location(whole,quest,d),POS(JJ,t))}{\|d\|_1}$	9.986
215.	$AvgNTF2WholeQuestJJ(R,t) = avg_{d \in R} \frac{TF(Location(whole,quest,d),POS(JJ,t))}{TF(d,t)}$	2.641
216.	$AvgNTFWholeQuestNN(R,t) = avg_{d \in R} \frac{TF(Location(whole,quest,d),POS(NN,t))}{\|d\|_1}$	**47.835**
217.	$AvgNTF2WholeQuestNN(R,t) = avg_{d \in R} \frac{TF(Location(whole,quest,d),POS(NN,t))}{TF(d,t)}$	9.919
218.	$AvgNTFWholeQuestNNP(R,t) = avg_{d \in R} \frac{TF(Location(whole,quest,d),POS(NNP,t))}{\|d\|_1}$	**15.163**
219.	$AvgNTF2WholeQuestNNP(R,t) = avg_{d \in R} \frac{TF(Location(whole,quest,d),POS(NNP,t))}{TF(d,t)}$	5.301
220.	$AvgNTFWholeExclaimJJ(R,t) = avg_{d \in R} \frac{TF(Location(whole,exclaim,d),POS(JJ,t))}{\|d\|_1}$	0.045
221.	$AvgNTF2WholeExclaimJJ(R,t) = avg_{d \in R} \frac{TF(Location(whole,exclaim,d),POS(JJ,t))}{TF(d,t)}$	2.088
222.	$AvgNTFWholeExclaimNN(R,t) = avg_{d \in R} \frac{TF(Location(whole,exclaim,d),POS(NN,t))}{\|d\|_1}$	7.955
223.	$AvgNTF2WholeExclaimNN(R,t) = avg_{d \in R} \frac{TF(Location(whole,exclaim,d),POS(NN,t))}{TF(d,t)}$	10.893
224.	$AvgNTFWholeExclaimNNP(R,t) = avg_{d \in R} \frac{TF(Location(whole,exclaim,d),POS(NNP,t))}{\|d\|_1}$	7.103
225.	$AvgNTF2WholeExclaimNNP(R,t) = avg_{d \in R} \frac{TF(Location(whole,exclaim,d),POS(NNP,t))}{TF(d,t)}$	10.798

Table 5.14: Sentence Type Characteristic - (3) Whole Document and POS Based

No.	Feature	Divergence
226.	$AvgNTFFirstStatJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,stat,d),POS(JJ,t))}{\|d\|_1}$	1.043
227.	$AvgNTF2FirstStatJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,stat,d),POS(JJ,t))}{TF(d,t)}$	2.289
228.	$AvgNTFFirstStatNN(R,t) = avg_{d\in R}\frac{TF(Location(first,stat,d),POS(NN,t))}{\|d\|_1}$	3.493
229.	$AvgNTF2FirstStatNN(R,t) = avg_{d\in R}\frac{TF(Location(first,stat,d),POS(NN,t))}{TF(d,t)}$	2.477
230.	$AvgNTFFirstStatNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,stat,d),POS(NNP,t))}{\|d\|_1}$	6.522
231.	$AvgNTF2FirstStatNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,stat,d),POS(NNP,t))}{TF(d,t)}$	8.622
232.	$AvgNTFFirstQuestJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,quest,d),POS(JJ,t))}{\|d\|_1}$	2.149
233.	$AvgNTF2FirstQuestJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,quest,d),POS(JJ,t))}{TF(d,t)}$	2.120
234.	$AvgNTFFirstQuestNN(R,t) = avg_{d\in R}\frac{TF(Location(first,quest,d),POS(NN,t))}{\|d\|_1}$	1.569
235.	$AvgNTF2FirstQuestNN(R,t) = avg_{d\in R}\frac{TF(Location(first,quest,d),POS(NN,t))}{TF(d,t)}$	10.382
236.	$AvgNTFFirstQuestNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,quest,d),POS(NNP,t))}{\|d\|_1}$	6.586
237.	$AvgNTF2FirstQuestNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,quest,d),POS(NNP,t))}{TF(d,t)}$	12.091
238.	$AvgNTFFirstExclaimJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,exclaim,d),POS(JJ,t))}{\|d\|_1}$	0.580
239.	$AvgNTF2FirstExclaimJJ(R,t) = avg_{d\in R}\frac{TF(Location(first,exclaim,d),POS(JJ,t))}{TF(d,t)}$	2.622
240.	$AvgNTFFirstExclaimNN(R,t) = avg_{d\in R}\frac{TF(Location(first,exclaim,d),POS(NN,t))}{\|d\|_1}$	0.808
241.	$AvgNTF2FirstExclaimNN(R,t) = avg_{d\in R}\frac{TF(Location(first,exclaim,d),POS(NN,t))}{TF(d,t)}$	9.477
242.	$AvgNTFFirstExclaimNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,exclaim,d),POS(NNP,t))}{\|d\|_1}$	**294.198**
243.	$AvgNTF2FirstExclaimNNP(R,t) = avg_{d\in R}\frac{TF(Location(first,exclaim,d),POS(NNP,t))}{TF(d,t)}$	5.477

Table 5.15: Sentence Type Characteristic - (4) Paragraph and POS Based <part 1>

No.	Feature	Divergence
244.	$AvgNTFBodyStatJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,stat,d),POS(JJ,t))}{\|d\|_1}$	0.589
245.	$AvgNTF2BodyStatJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,stat,d),POS(JJ,t))}{TF(d,t)}$	1.581
246.	$AvgNTFBodyStatNN(R,t) = avg_{d\in R}\frac{TF(Location(body,stat,d),POS(NN,t))}{\|d\|_1}$	2.215
247.	$AvgNTF2BodyStatNN(R,t) = avg_{d\in R}\frac{TF(Location(body,stat,d),POS(NN,t))}{TF(d,t)}$	8.966
248.	$AvgNTFBodyStatNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,stat,d),POS(NNP,t))}{\|d\|_1}$	3.404
249.	$AvgNTF2BodyStatNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,stat,d),POS(NNP,t))}{TF(d,t)}$	10.886
250.	$AvgNTFBodyQuestJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,quest,d),POS(JJ,t))}{\|d\|_1}$	**38.309**
251.	$AvgNTF2BodyQuestJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,quest,d),POS(JJ,t))}{TF(d,t)}$	2.307
252.	$AvgNTFBodyQuestNN(R,t) = avg_{d\in R}\frac{TF(Location(body,quest,d),POS(NN,t))}{\|d\|_1}$	0.192
253.	$AvgNTF2BodyQuestNN(R,t) = avg_{d\in R}\frac{TF(Location(body,quest,d),POS(NN,t))}{TF(d,t)}$	2.490
254.	$AvgNTFBodyQuestNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,quest,d),POS(NNP,t))}{\|d\|_1}$	**67.227**
255.	$AvgNTF2BodyQuestNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,quest,d),POS(NNP,t))}{TF(d,t)}$	8.816
256.	$AvgNTFBodyExclaimJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,exclaim,d),POS(JJ,t))}{\|d\|_1}$	5.459
257.	$AvgNTF2BodyExclaimJJ(R,t) = avg_{d\in R}\frac{TF(Location(body,exclaim,d),POS(JJ,t))}{TF(d,t)}$	0.647
258.	$AvgNTFBodyExclaimNN(R,t) = avg_{d\in R}\frac{TF(Location(body,exclaim,d),POS(NN,t))}{\|d\|_1}$	4.774
259.	$AvgNTF2BodyExclaimNN(R,t) = avg_{d\in R}\frac{TF(Location(body,exclaim,d),POS(NN,t))}{TF(d,t)}$	12.285
260.	$AvgNTFBodyExclaimNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,exclaim,d),POS(NNP,t))}{\|d\|_1}$	0.429
261.	$AvgNTF2BodyExclaimNNP(R,t) = avg_{d\in R}\frac{TF(Location(body,exclaim,d),POS(NNP,t))}{TF(d,t)}$	11.514

Table 5.16: Sentence Type Characteristic - (4) Paragraph and POS Based <part 2>

No.	Feature	Divergence
262.	$AvgNTFLastStatJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,stat,d),POS(JJ,t))}{\|d\|_1}$	5.514
263.	$AvgNTF2LastStatJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,stat,d),POS(JJ,t))}{TF(d,t)}$	2.622
264.	$AvgNTFLastStatNN(R,t) = avg_{d \in R} \frac{TF(Location(last,stat,d),POS(NN,t))}{\|d\|_1}$	**<u>468.493</u>**
265.	$AvgNTF2LastStatNN(R,t) = avg_{d \in R} \frac{TF(Location(last,stat,d),POS(NN,t))}{TF(d,t)}$	9.395
266.	$AvgNTFLastStatNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,stat,d),POS(NNP,t))}{\|d\|_1}$	**<u>15.130</u>**
267.	$AvgNTF2LastStatNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,stat,d),POS(NNP,t))}{TF(d,t)}$	5.482
268.	$AvgNTFLastQuestJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,quest,d),POS(JJ,t))}{\|d\|_1}$	0.141
269.	$AvgNTF2LastQuestJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,quest,d),POS(JJ,t))}{TF(d,t)}$	1.471
270.	$AvgNTFLastQuestNN(R,t) = avg_{d \in R} \frac{TF(Location(last,quest,d),POS(NN,t))}{\|d\|_1}$	2.192
271.	$AvgNTF2LastQuestNN(R,t) = avg_{d \in R} \frac{TF(Location(last,quest,d),POS(NN,t))}{TF(d,t)}$	9.532
272.	$AvgNTFLastQuestNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,quest,d),POS(NNP,t))}{\|d\|_1}$	0.563
273.	$AvgNTF2LastQuestNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,quest,d),POS(NNP,t))}{TF(d,t)}$	10.049
274.	$AvgNTFLastExclaimJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,exclaim,d),POS(JJ,t))}{\|d\|_1}$	0.001
275.	$AvgNTF2LastExclaimJJ(R,t) = avg_{d \in R} \frac{TF(Location(last,exclaim,d),POS(JJ,t))}{TF(d,t)}$	2.293
276.	$AvgNTFLastExclaimNN(R,t) = avg_{d \in R} \frac{TF(Location(last,exclaim,d),POS(NN,t))}{\|d\|_1}$	0.559
277.	$AvgNTF2LastExclaimNN(R,t) = avg_{d \in R} \frac{TF(Location(last,exclaim,d),POS(NN,t))}{TF(d,t)}$	2.508
278.	$AvgNTFLastExclaimNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,exclaim,d),POS(NNP,t))}{\|d\|_1}$	**<u>50.291</u>**
279.	$AvgNTF2LastExclaimNNP(R,t) = avg_{d \in R} \frac{TF(Location(last,exclaim,d),POS(NNP,t))}{TF(d,t)}$	8.877

Table 5.17: Sentence Type Characteristic - (4) Paragraph and POS Based <part 3>

5.3.5 Title Term Characteristic

Title term characteristic are the features related to the relationship between the query term and the title query terms. The features in the title term characteristic are shown in Table 5.18. These features can be divided into three types: (1) 'Dist2TitleInPara' is the average distance between the query term and each title query term in each paragraph of a relevant document, (i.e., Features 280 and 281); (2) 'TitlePairFreqInPara' is the average number of pairs of query terms and title query terms in each paragraph of a particular document (i.e., Features 282 and 283); and (3) 'ParaFreqHaveTitlePair' is the number of paragraphs that have both query term and any one of title query terms (i.e., Features 284 and 285). The equations for calculating Types (1), (2) and (3) features are shown in Equation 5.4, 5.5 and 5.6, respectively. The $Position(Location(p, whole, d), t)$ is a list of positions of the given term t that appears in the paragraph p of the document d, and the $avg_{p \in d}(.)$ is equal to the $\sum_{p \in d}(.)$ divided by the number of paragraphs in the document d. The $abs(.)$ function returns

No.	Feature	Divergence
280.	$SumDist2TitleInPara(R,t) = \sum_{d \in R} avg_{p \in d} Dist2Title(Location(p, whole, d), t, T)$	0.855
281.	$AvgDist2TitleInPara(R,t) = \frac{SumDist2TitleInPara(R,t)}{card(R)}$	0.182
282.	$SumTitlePairFreqInPara(R,t) = \sum_{d \in R} avg_{p \in d} PairFreq(Location(p, whole, d), t, T)$	0.813
283.	$AvgTitlePairFreqInPara(R,t) = \frac{SumTitlePairFreqInPara(R,t)}{card(R)}$	0.189
284.	$SumParaFreqHaveTitlePair(R,t) = \sum_{d \in R} avg_{p \in d} HavePair(Location(p, whole, d), t, T)$	0.985
285.	$AvgParaFreqHaveTitlePair(R,t) = \frac{SumParaFreqHaveTitlePair(R,t)}{card(R)}$	0.915

Table 5.18: Title Term Characteristic

the absolute value. In addition, the 'Sum' and 'Avg' intra-document normalization methods are also used to normalize the term frequency:

$$\begin{aligned} Dist2Title(Location(p, whole, d), t, T) = \; & \textstyle\sum_{W_t \in Position(Location(p,whole,d),t)} \sum_{k \in T} \\ & \sum_{W_k \in Position(Location(p,whole,d),k)} abs(W_t - Wk) \end{aligned} \tag{5.4}$$

$$\begin{aligned} PairFreq(Location(p, whole, d), t, T) = \; & TF(Location(p, whole, d), t) \times \\ & \sum_{k \in T} TF(Location(p, whole, d), k) \end{aligned} \tag{5.5}$$

$$\begin{cases} HavePair(Location(p, whole, d), t, T) \equiv 0 & \text{for } PairFreq(Location(p, whole, d), t, T) = 0 \\ HavePair(Location(p, whole, d), t, T) \equiv 1 & \text{for } PairFreq(Location(p, whole, d), t, T) > 0 \end{cases} \tag{5.6}$$

From Table 5.18 we can see that all of the features achieve poor divergence values. This reveals that title terms characteristic is not good for classification.

5.3.6 Features used in GQTE

The features used for the classification module in GQTE are determined by the divergence value of the features. According to the divergence values of each feature shown in Tables 5.2 to 5.18, we realize that there are many features that achieve lower than 15 divergence value. Therefore, we choose 15 as the threshold value for determining the features used in GQTE. Table 5.19 shows the summary of the features used for the classification module in GQTE. These features are bold-faced and underlined in Tables 5.2 to 5.18.

No.	Feature	Divergence
1.	$AvgNTFLastStatNN(R,t)$	468.493
2.	$AvgNTFFirstExclaimNNP(R,t)$	294.198
3.	$CHI_3 * NSumNTF(t)$	133.984
4.	$AvgNTFBodyQuestNNP(R,t)$	67.227
5.	$AvgNTFLastWholeNN(R,t)$	59.418
6.	$AvgNTF2FirstFirstNN(R,t)$	56.458
7.	$AvgNTFLastExclaimNNP(R,t)$	50.291
8.	$AvgNTFFirstWholeNN(R,t)$	48.437
9.	$AvgNTFWholeQuestNN(R,t)$	47.835
10.	$MinNTF(C,t)$	44.756
11.	$AvgNTFBodyQuestJJ(R,t)$	38.309
12.	$AvgNTF2FirstFirstJJ(R,t)$	34.054
13.	$W4 * NMaxNTF(t)$	31.576
14.	$NSumNTF(t)$	29.342
15.	$AvgNTFBodyWholeNN(R,t)$	26.658
16.	$AvgNTF2BodyLastNN(R,t)$	25.081
17.	$AvgNTFFirstWhole(R,t)$	19.234
18.	$AvgNTFWholeQuest(R,t)$	18.971
19.	$AvgNTFLastWhole(R,t)$	18.698
20.	$MinNTF(R,t)$	18.291
21.	$SumNTFBodyWhole(R,t)$	15.672
22.	$NMaxNTF(t)$	15.600
23.	$AvgNTFFirstWholeNNP(R,t)$	15.325
24.	$AvgNTFWholeQuestNNP(R,t)$	15.163
25.	$AvgNTFLastWholeNNP(R,t)$	15.152
26.	$AvgNTFLastStatNNP(R,t)$	15.130

Table 5.19: A summary of the features used for the classification module in GQTE

5.4 Implementation of the GQTE

The implementation of the GQTE is described in this section. Firstly, the learning procedure of the classification module in GQTE is introduced, which uses a new method, cross training method, to train the classifier based on different collections. The accuracy of the classifier is estimated based on a 10-fold cross-validation. Moreover, some practices in GQTE are presented and evaluated based on the estimation accuracy and the retrieval effectiveness. Several methods for generating the new queries and merging the retrieval lists are also investigated and examined.

5.4.1 Learning in GQTE

In order to investigate the impact on different collections, such as TREC-6, 7 and 8, we propose to train and test the dataset by combining the collections in turn rather than by a single collection only. This method is called cross training in our study.

Cross training is a supervised, multi-collection learning method based on the C4.5 classification method. This method is designed for training the dataset that contains different data collections. The dataset D is split into c subset. Each subset contains one collection. The classifier is trained $\sum_{r=1}^{c} \frac{c!}{r!(c-r)!}$ times. Each time, the classifier is trained on r subsets and tested on the remaining $(c-r)$ subsets. The accuracy of each induced classifier is estimated by using 10-fold cross-validation, where the induced classifier is generated using 90% of the dataset as training set and the remaining 10% as test set repeating ten times with changing training and test sets. The accuracy value is calculated by two measures: precision and recall. The precision is computed by the total number of the correct classifications and then divided by the number of the estimated 'Good' query terms for a given topic, whereas the recall is calculated by the total number of the correct classifications and then divided by the number of terms in the near optimal query (i.e., the query obtained in Chapter 3 or Chapter 4) for a given topic. The correct classification means that the estimated 'Good' query term is a near optimal query term of that

No.	Data subset	No. of queries	No. of ‘Good’ cases	No. of ‘Normal’ cases
1.	TREC-6	50	3315	3315
2.	TREC-7	50	3648	3648
3.	TREC-8	50	3800	3800

Table 5.20: Four datasets for training the classifier in GQTE

topic. It is be noticed that the correct classifications of ‘Normal’ query term are not calculated.

Table 5.20 shows the detailed information of the dataset used in this training. The dataset contains three subsets, TREC-6, TREC-7 and TREC-8, corresponding to the collections of each TREC. In particular, each data subset consists of an equal number of ‘Good’ cases and ‘Normal’ cases. The ‘Good’ cases are the query terms in the best near optimal queries that are the best among all the near optimal queries for each topic in Chapters 3 and 4, whereas the ‘Normal’ cases are randomly chosen from the remaining terms in the relevant documents for that topic. The features are those good features shown in Table 5.19.

The detailed training process and the accuracy of the induced classifier are shown in Table 5.21. From the table we can see that the precision and the recall of all the training processes are higher than 0.930 and 0.87, respectively. Moreover, the average precision and recall, which is calculated by the average of the precision values and the recall values among all the processes, is about 0.950 and 0.943, respectively. These reveal that the accuracy of our classifier is good. Moreover, the precision and recall of using training sets containing the TREC-6 collection (i.e., Nos. 1, 4, 5 and 7) perform better than other collections. This may be due to the fact that the TREC-6 collection contains more documents (i.e., Congressional Records documents), than TREC-7 and TREC-8. Furthermore, we can see that the precision and recall of the training set containing all the collections (i.e. No. 7) are the highest among others. Therefore, the induced classifier based on training set No. 7 is chosen as the classification module in GQTE.

No.	Training Set	Test Set 1		Test Set 2	
		Prec	Recall	Prec	Recall
1.	TREC-6	TREC-7		TREC-8	
		0.932	0.972	0.938	0.976
2.	TREC-7	TREC-6		TREC-8	
		0.951	0.884	0.953	0.968
3.	TREC-8	TREC-6		TREC-7	
		0.951	0.870	0.957	0.945
4.	TREC-6, 7	TREC-8		-	
		0.946	0.977		
5.	TREC-6, 8	TREC-7		-	
		0.949	0.962		
6.	TREC-7, 8	TREC-6		-	
		0.954	0.885		
7.	TREC-6, 7, 8	TREC-6, 7, 8		-	
		0.958	0.972		
Average Precision				0.950	
Average Recall				0.943	

Table 5.21: The cross training process of the dataset containing TREC-6, 7 and 8

5.4.2 Practice in GQTE

It is a fact that the relevant documents about regarding the user's information needs are unknown during the retrieval process in the real world. However, most of the features used in the classification module of GQTE are calculated based on the relevant documents set. In order to fulfil this criterion in practice, we assume that there are only 10 documents related to the given user's information needs in the collection, and that all of these documents can be retrieved in the top 10 of the given title query. This is similar to the assumption in Pseudo Relevance Feedback, and the number is determined by the observations in Section 3.6 of Chapter 3. Therefore, all the terms in the top 10 retrieved documents of the title query can be estimate the class by the classification module in GQTE. The estimated 'Good' query terms are then used to reformulate the title query by Equation 5.3. The number of estimated 'Good' query terms added is decided by the confidence level, and the query term weight is determined by the term weight scheme shown in Table 5.1.

Evaluation on estimation accuracy

In order to investigate the effects on the estimation accuracy of the different confidence levels, seven confidence levels: 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.98 are examined and are shown in Figure 5.2. In the figure, the Y-axis on the left hand side is used for the precision and recall, whereas the Y-axis on the right hand side is used for the number of the estimated 'Good' query terms. This precision and recall are calculated similarly to the previous section. The precision is computed by the total number of the correct classifications and then divided by the number of the estimated 'Good' query terms for a given topic, whereas the recall is calculated by the total number of the correct classifications and then divided by the number of terms in the near optimal query (i.e., the query obtained in Chapter 3 or Chapter 4) for a given topic. The correct classification means that the estimated 'Good' query term is the near optimal query term of that topic. This experiment is carried out for the TREC-6, TREC-7 and TREC-8 test collections.

From Figure 5.2, we can see that when the confidence level is increased, the precision will increase, whereas the recall will decrease and the number of terms will decrease as expected. Moreover, it is obvious that the performance of TREC-7 and TREC-8 is similar. This may be due to the fact that the corpus in TREC-7 and TREC-8 are the same. Furthermore, we realize that all of the precision, recall and the number of terms will increase or will decrease significantly when the confidence level changes from 0.95 to 0.98. This implies that the confidence level at 0.95 may be a good threshold value for determining the query size. Finally, we can see that both of the precision and recall are smaller than 0.1; This is very poor. It implies that there is a big difference of the characteristic between the term features of using the top 10 retrieved documents of the title query and the term features of using the entire relevant documents set, so that the top 10 retrieved documents of the title query cannot be used to simulate the entire relevant document set for classification. In fact, the top 10 retrieved documents of title query are only a small subset of the

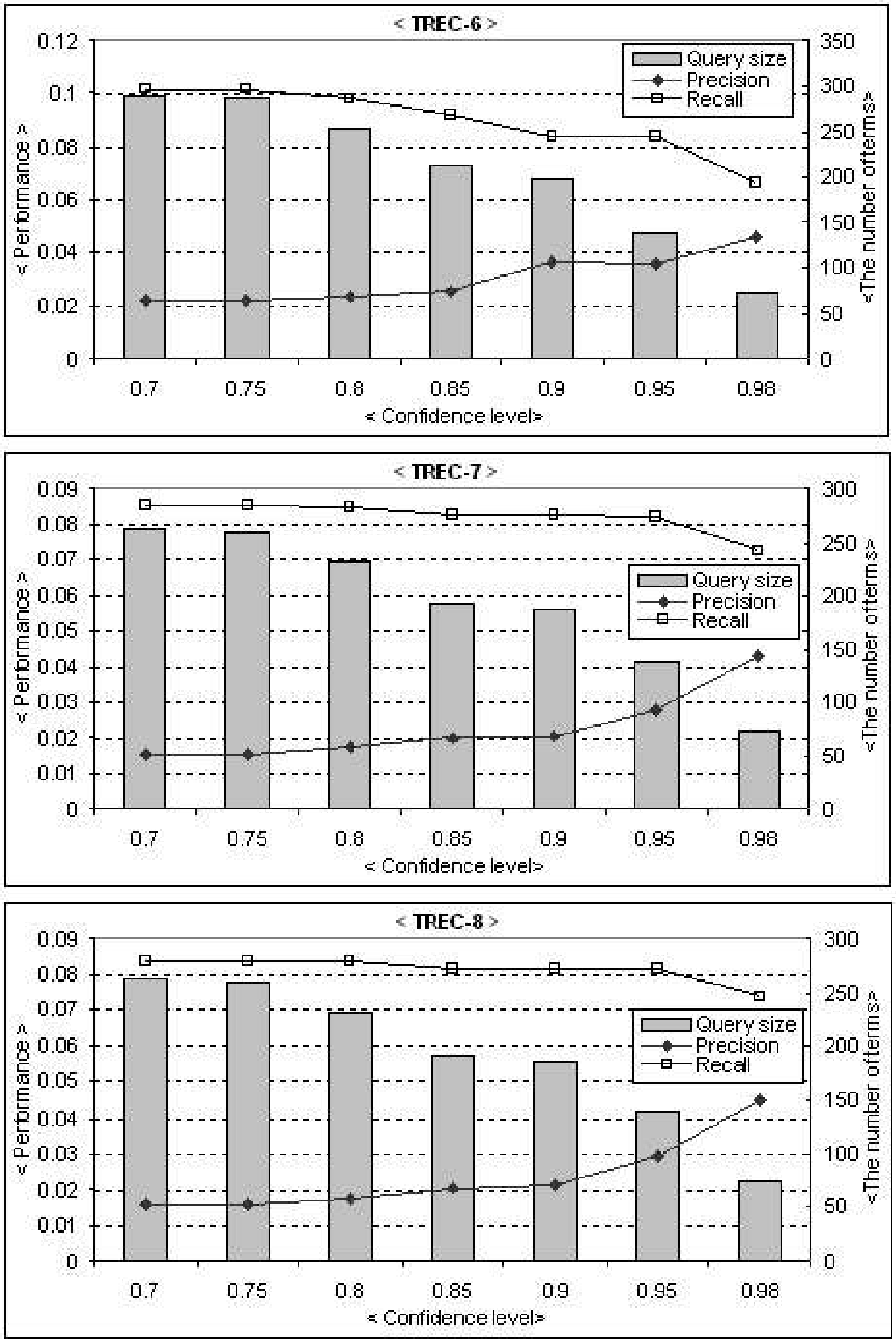


Figure 5.2: The performance of the classification module in GQTE against the different confidence levels

Collection	Precision	Recall
TREC-6	0.396	0.132
TREC-7	0.428	0.088
TREC-8	0.472	0.113

Table 5.22: The average precision and recall of the top 10 retrieved documents of the title queries

entire relevant document set. Table 5.22 shows the average precision and recall of the top 10 retrieved documents of the title queries in TREC-6, 7 and 8 collections, respectively, where the precision and recall are calculated by the total number of relevant documents in the top 10 retrieved documents divided by the number of all the relevant documents for the topic and the number of retrieved documents (i.e., 10), respectively. From the table we can see that the precision and recall are lower than 0.5 and 0.2, respectively. This emphasizes that using the top 10 retrieved documents of the title query to simulate the entire relevant documents set is inaccurate.

Evaluation on Retrieval Effectiveness

In order to investigate the effects of retrieval effectiveness of the different term weight schemes and the different confidence levels, four term weight scheme are shown in Table 5.1, with the same seven confidence levels examined and shown in Figures 5.3, 5.4 and 5.5. In particular, Figure 5.3 shows the MAP values of the reformulated queries, whereas Figure 5.4 shows the R-prec values and Figure 5.5 shows the P@30 values. Four term weight schemes in Table 5.1 are labeled in sequence: (1) ‘unity’ refers to the unity weight scheme; (2) ‘rank’ refers to the rank normalization scheme; (3) ‘rank2’ refers to the rank normalization 2 scheme; and (4) ‘conf’ refers to the confidence scheme. In addition, the label ‘Title’ refers to the retrieval effectiveness of the Title query, which can be used for the baseline performance. This experiment is carried out for the TREC-6, TREC-7 and TREC-8 test collections, and our VSM is used.

There are some interesting results in this experiment. First, the ‘rank2’ term weight scheme achieves highest retrieval effectiveness for all the measures (i.e., MAP, R-prec and P@30) in all the test collections. This value is even higher than the retrieval effectiveness of the title query (i.e., labeled ‘Title’). Therefore, we believe that our GQTE can improve the retrieval effectiveness of the user query. Second, some of the term weight schemes achieve lower retrieval effectiveness than the title query. This reveals that a suitable choice of weights can improve the performance. Third, similar to the results shown in Figure 5.2, the performance of TREC-7 and TREC-8 is alike. Therefore, we believe that our GQTE will achieve similar retrieval effectiveness for the same corpus. Fourth, by comparing with the title query, the P@30 performance is improved most among all the measures. This implies that our GQTE is good at pushing up the relevant documents in the retrieval list rather than excluding the irrelevant documents from the retrieval list. Finally, it is obvious that the retrieval effectiveness of using our GQTE is still far below the retrieval effectiveness of the near optimal queries (Table 4.1 in Chapter 4). We believe that there are two possible causes of low retrieval effectiveness; first, we improperly used the top retrieved documents of title query to simulate the entire relevant document set; Second, we improperly filtered out the relevant documents in the initial retrieval list of the title query.

In order to further improve the retrieval effectiveness of the GQTE, four different merge list methods are proposed and are shown in Table 5.23. These methods are used to reorganize the new retrieval list B with retrieval list A of the title query. In the table, $A \cap B$ is a set of the documents that appears in both the retrieval list A and B, and the value ‘0’, ‘1’ and ‘2’ are the weights used for ranking. The rank of the documents in the merged retrieval list is computed by the similarity score of the document in the original retrieval list, divided by the maximum similarity score in the original retrieval list, and then multiple by the weights (i.e., 0, 1 or 2 or simply fix the document). In particular, ‘Fix A’ method returns the same retrieval list A

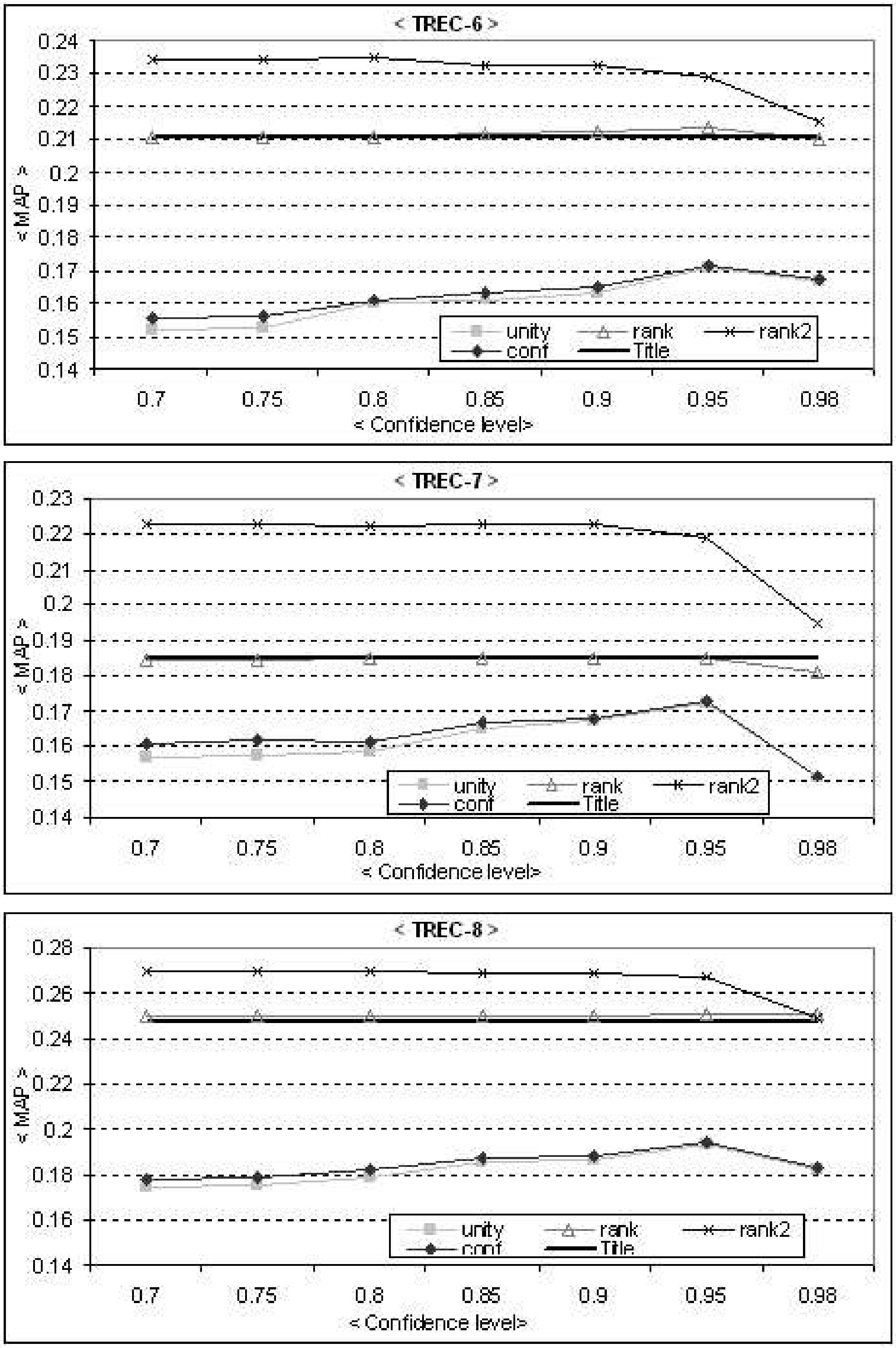


Figure 5.3: The MAP of the GQTE

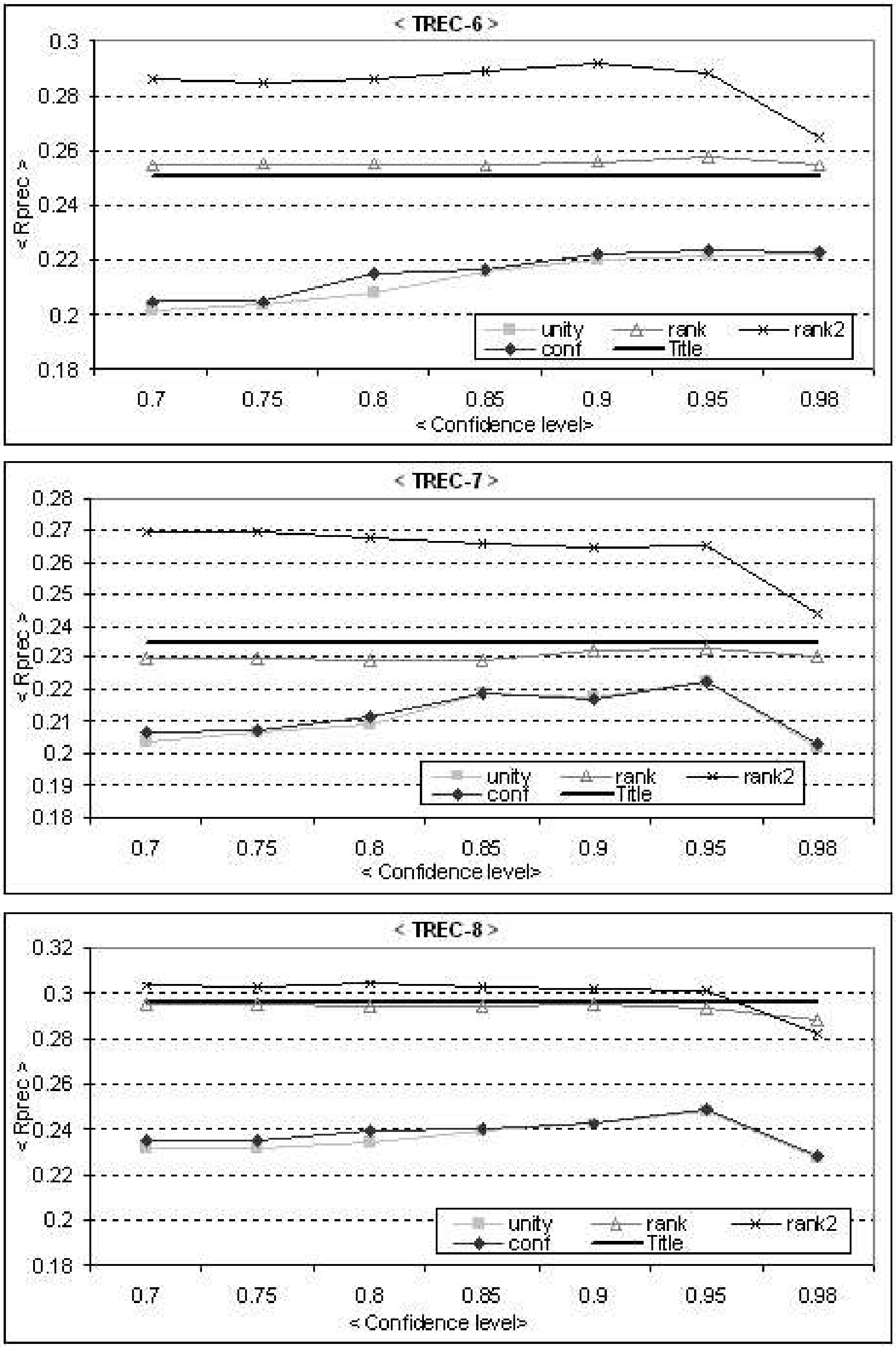


Figure 5.4: The R-prec of the GQTE

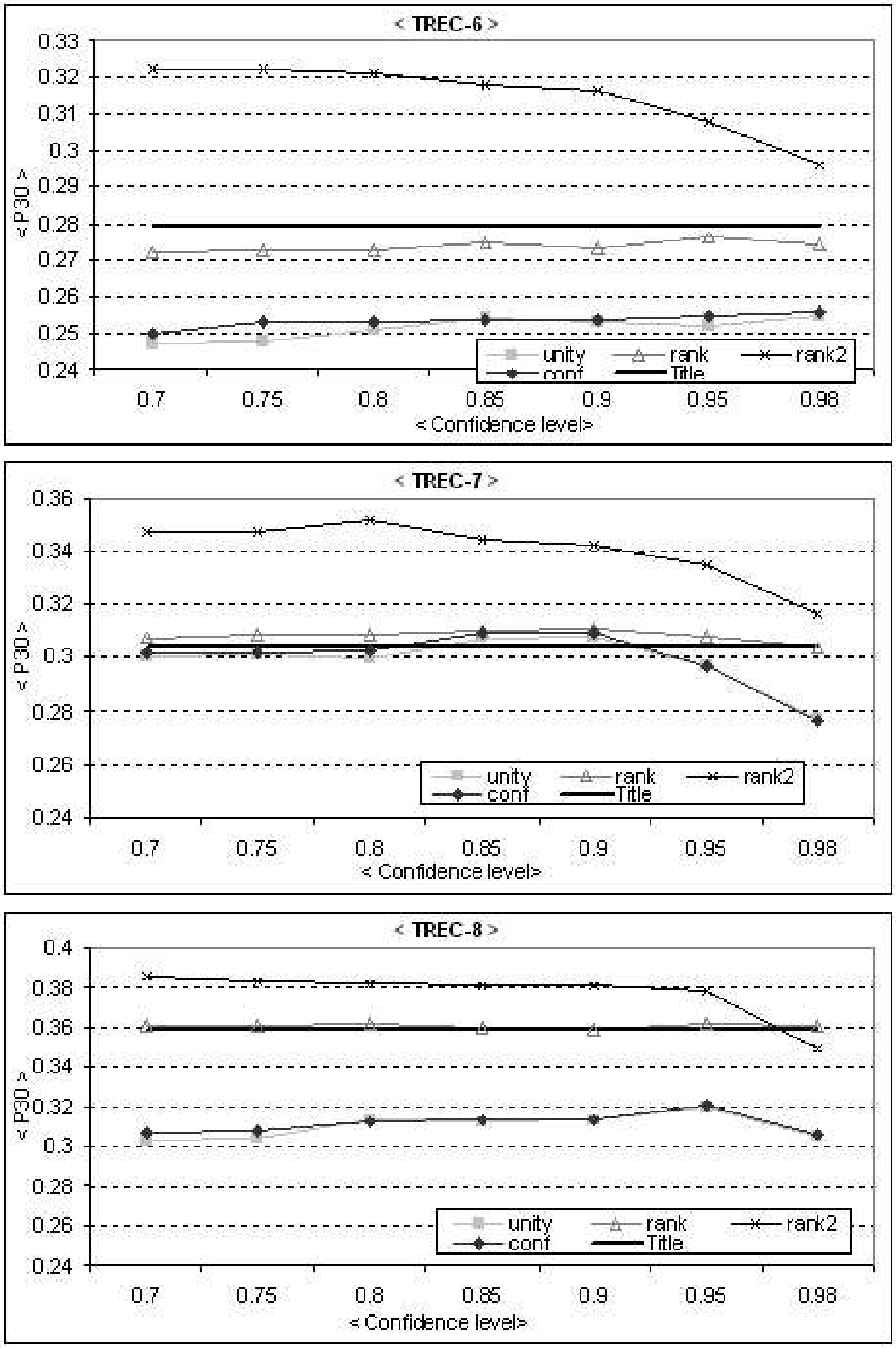


Figure 5.5: The prec@30 of the GQTE

No.	Method Name	Weights on $(A \cap B)$	Weights on A	Weights on B
1	Fix A	fix at top	1	0
2	Fix B	fix at top	0	1
3	Highly A	2	1	0
4	Highly B	2	0	1
5	Combined	3	top 10*4, other*1	2

Table 5.23: Merge list method for reorganizing the current retrieval list A with the retrieval list of the title query B

but pushes the documents, which appear in $A \cap B$, to the top. 'Fix B' method is similar to 'Fix A' method but keeps the B rather than A. Moreover, 'Highly A' and 'Highly B' methods push up the documents in $A \cap B$ according to the weights rather than simply pushing up to the top of the retrieval list. The last method, 'Combined' method, does not only push up the documents in $A \cap B$, but also pushes up the top 10 retrieved documents of the title query. This is designed for considering the rank freeze effects. This experiment is carried out by using the 'rank 2' term weight scheme with two different confidence levels: 0.8 and 0.9.

Figure 5.6 shows the retrieval effectiveness of different merge list methods against two confidence levels and three test collections. In the figure, the labels 'Title' and 'rank2' refer to the retrieval effectiveness of the title query and the query reformulated by using 'rank2' term weight scheme, respectively. These two datasets can be used for the baseline performance. From the figure we can see that most of the merge list methods perform more poorly than the 'rank2' for all the measures (i.e., MAP, R-prec and P@30) in all the test collections except the 'Combined' merge list method. The 'Combined' method performs better MAP and R-prec values only in the TREC-8 collection. It reveals that our merge list methods cannot significantly improve the retrieval effectiveness. Moreover, we realize that the merge list method based on the new retrieval list (i.e., B) performs better than based on the retrieval list of the 'title' query (i.e., 'Fix B' and 'Highly B' method performs better). Finally, we can see that 'Highly A' and 'Highly B' methods are performed better than 'Fix A' and 'Fix B' methods, respectively. This illustrates that pushing up the documents by weights is better than pushing up the documents to the top.

5.5 Summary

The contribution of this chapter is to propose a novel query term expansion method, Good Query Term Extractor (GQTE). The GQTE can be used to reformulate the user query with the estimated 'Good' query terms. The experimental results tell us that this method is a useful method to enhance the query retrieval effectiveness, but the improvement is still far below the expected improvement (i.e., similar to the retrieval effectiveness of the near optimal queries). We believe that a suitable choice of the retrieved documents can substantially enhance the retrieval effectiveness of the GQTE. Moreover, a novel term weight scheme is proposed which uses the confidence value of the estimation to calculate the weights and ranks. This scheme performs better than other term weight schemes. Furthermore, a new method based on the confidence level to decide the number of terms added to the user query is introduced. Finally, many characteristics of the near optimal queries are investigated. These will be helpful in understanding what kind of query terms will perform better.

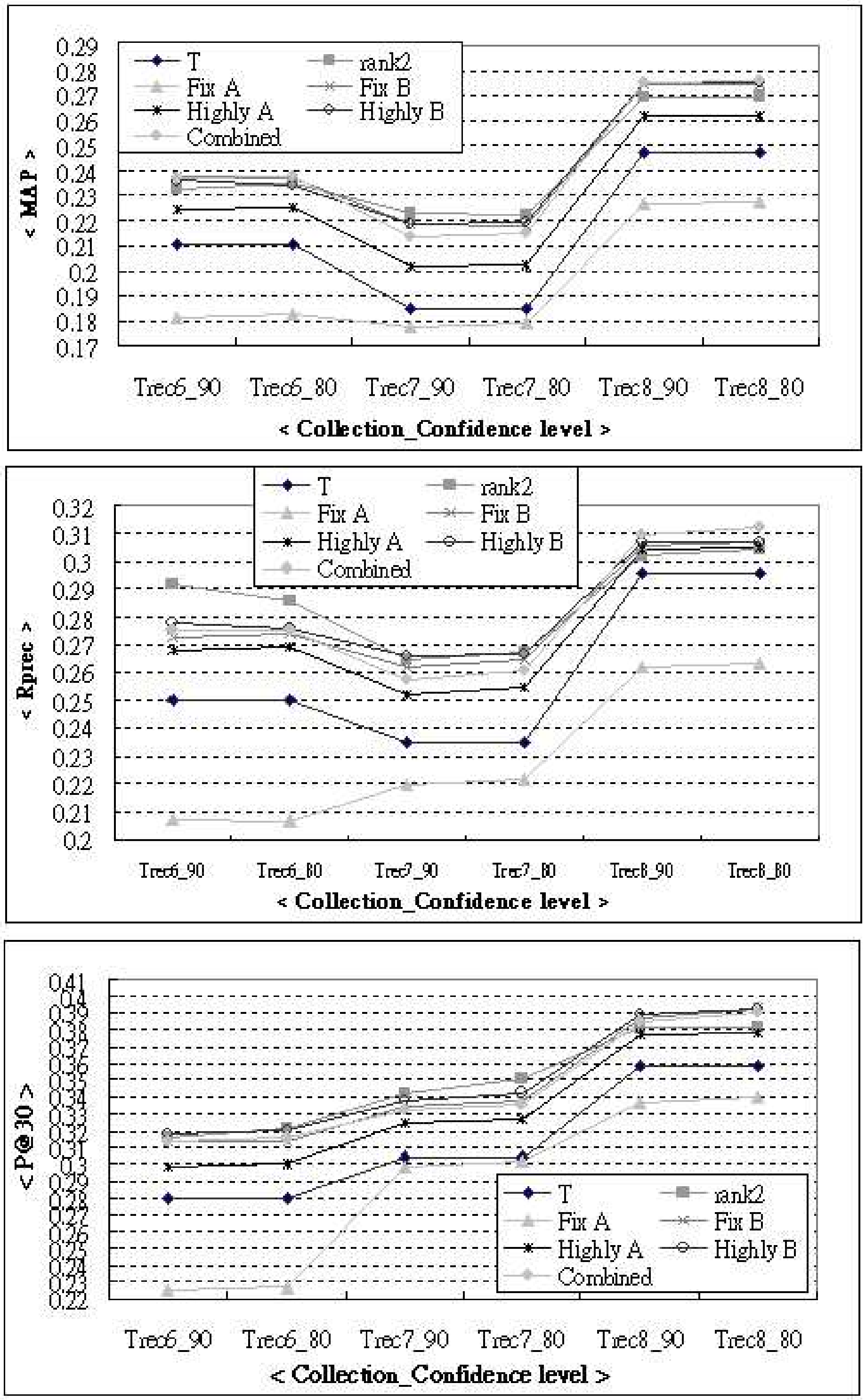


Figure 5.6: The retrieval effectiveness of different merge list methods against two confidence levels and three test collections

Chapter 6

Conclusions and Future Work

The contribution of this study is to illustrate that the optimal retrieval effectiveness as well as the near optimal queries can be obtained in existing IR systems. We confirm this in our experiments using three TREC test collections for English ad hoc retrieval. In this dissertation we presented a systematic and practical study on how to find and estimate the near optimal queries based on two strategies, IRF and Combinatorial Optimization Search. Based in the observations on this study, we proposed a novel query term expansion method, Good Query Term Extractor (GQTE), to reformulate the user query with the estimated 'Good' query terms. From our experimental results, the following major conclusions can be drawn:

- Most of the queries achieved either good or poor MAP values independent of the use of different retrieval models. This finding reveals that the main cause of the poor retrieval effectiveness in existing IR systems may not be the system factor. Moreover, most of the near optimal queries about the topics in test collections have been found under some idealized situations. It appears that a suitable choice of terms and a suitable choice of weights can substantially enhance the retrieval effectiveness in existing IR systems.
- An IRF method is proposed to find the near optimal query. This method is the Relevance Feedback method in idealized situations. The term 'idealized' means that some of the relevance feedback constraints have been relaxed or

some of the practical limitations have been overcome. Our IRF can be implemented by using a perfect classifier that identifies all relevant documents from the entire retrieval list of long query (i.e., TDN query) without errors. If we use title query rather than long query, the optimal retrieval effectiveness may not be achieved. This suggests that for title queries RF needs at least two iterations in order to ensure that the retrieval list has an adequately high recall to obtain the potential IRF performance.

- The performance difference between the better term ranking functions in IRF actually does not differ substantially using the log-odds ratio of Robertson and Karen Sparck Jones, the Chi-square, Kullback-Leibler divergence, or the RSV by Robertson, provided either the average normalized term frequency or the average of the maximum of the normalized term frequencies are used.

- We have experimented with our IRF and our GQTE on weighted query. Our best term weight scheme is to assign the term weight based on the rank of the individual term. This rank can be determined by the term ranking function or by the GQTE's confidence value. In addition, the experimental results illustrate that a suitable choice of negative terms can enhance the retrieval effectiveness.

- Our best MAP value of the various query sizes for different topics in IRF is not significantly better than that achieved using the fixed query size. This reveals that various query size for different topics may not be better. Our best fixed query size was 100, 175 and 250 for unweighted terms, and positively weighted terms and negatively weighted terms, respectively. In addition, the query size can be determined by the confidnece value of the estimation when using our GQTE.

- A Combined Search method is proposed that can be used to find the near optimal query faster than our Simulated Annealing method. This method is a combination of Best First Search and Simulated Annealing Search. The retrieval effectiveness of our Combined Search is better than the local search methods, which have been implemented in this study (i.e., Hill Climbing Search, Best First Search and Simulated Annealing Search), and is also better than our IRF.

- Many characteristics of the near optimal query terms have been investigated.

Table 6.1 shows a summary of our best near optimal queries obtained in this study. We can see that the combine search method is the best method for all the measures (i.e., the MAP, the R-prec and the query size). In addition, although the performance of our GQTE is poor, it is better than using the original title query. This reveals that GQTE is also an effective method to expand the user query.

This research can be extended in several ways, such as:

1. Many topics still cannot obtain optimal retrieval effectiveness by using our combine search method. One of the possible reasons is that the near optimal query should have a weight for each term. The other possible reason is that the near optimal query should use the negative terms to discriminate the non-relevant documents from the collection.

2. There is still a lot of room to enhance our GQTE. One of the possible future directions is to enhance the estimation accuracy of the entire relevant documents set. We think that a well-trained relevant document classifier or relevant feedback with user judgement for the top retrieved documents may be an effective method.

Proposed Method	TREC-6	TREC-7	TREC-8	TREC-2005
Average MAP				
Original title query	0.211	0.185	0.247	0.173
Original long query	0.231	0.236	0.273	-
IRF with unweighted terms	0.616	0.592	0.556	0.546
IRF with positively weighted terms	0.640	0.599	0.574	0.606
IRF with positively and negatively weighted terms	0.637	0.624	0.591	0.630
Combine Search	0.726	0.755	0.752	0.711
GQTE	0.232	0.223	0.269	-
Average R-prec				
Original title query	0.250	0.235	0.296	0.237
Original long query	0.268	0.268	0.321	-
IRF with unweighted terms	0.614	0.582	0.547	0.543
IRF with positively weighted terms	0.625	0.582	0.561	0.582
IRF with positively and negatively weighted terms	0.619	0.603	0.572	0.607
Combine Search	0.774	0.785	0.785	0.743
GQTE	0.292	0.265	0.302	-
Average query size				
Original title query	2	2	2	3
Original long query	57	37	37	66
IRF with unweighted terms	100	105	88	100
IRF with positively weighted terms	173	172	190	175
IRF with positively and negatively weighted terms	165	165	174	369
Combine Search	66	68	71	50
GQTE	198	185	186	-

Table 6.1: Summary of our best performance

Bibliography

[AF77] R. Attar and A. S. Frankel. Local feedback in full-text retrieval systems. *Journal of the ACM*, 24(3):397–417, 1977.

[All96] J. Allan. Incremental Relevance Feedback for Information Filtering. In *ACM SIGIR'96*, pages 270–278, 1996.

[BAS93] C. Buckley, J. Allan, and G. Salton. Automatic routing and ad hoc retrieval using SMART: TREC 2. In *TREC-2, NIST Special Publication*, pages 45–56, 1993.

[BBM02] H. Billhardt, D. Borrajo, and V. Maojo. Using Genetic Algorithms to Find Suboptimal Retrieval Expert Combination. In *ACM Symposium on Applied computing*, pages 657–662, 2002.

[BH03] C. Buckley and D. Harman. Reliable Information Access Final Workshop Report. In *RIA, NRRC*, pages 1–30, 2003.

[BJN00] A. Bergstrom, P. Jaksetic, and P. Nordin. Enhancing Information Retrieval by Automatic Acquisition of Textual Relations using Genetic Programming. In *ACM International Conference on Intelligent user interfaces*, pages 29–32, 2000.

[CH67] T. M. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[CL04] Y. M. Chung and J. Y. Lee. Optimization of some factors affecting the performance of query expansion. In *Information Processing and Management, in press*, 2004.

[CMRB01] C. Carpineto, R. D. Mori, G. Romano, and B. Bigi. An Information-Theoretic Approach to Automatic Query Expansion. *ACM Transactions on Information Systems*, 19(1):1–27, 2001.

[CR02] C. Carpineto and G. Romano. Improving retrieval feedback with multiple term-ranking function combination. *ACM Transactions on Information Systems*, 20(3):259–290, 2002.

[DHB90] T. Dietterich, H. Hild, and G. Bakiri. A Comparative Study of ID3 and Back propagation for English Text-to-Speech Mapping. In *Seventh Intl. Conference on Machine Learning*, pages 24–31, 1990.

[Dun97] M. D. Dunlop. The effect of accessing nonmatching documents on relevance feedback. *ACM Transactions on Information Systems*, pages 137–153, 1997.

[Eft93] E. N. Efthimiadis. A user-centred evaluation of ranking algorithms for interactive query expansion. In *ACM SIGIR'93*, pages 148–159, 1993.

[FGP00] W. Fan, M. D. Gordon, and P. Pathak. Personalization of search engine services for effective retrieval and knowledge management. In *Proceedings of the Twenty First International Conference on Information Systems*, pages 20–34, 2000.

[FGP05] W. Fan, M. Gordon, and P. Pathak. Genetic Programming-Based Discovery of Ranking Functions for Effective Web Search. *Journal of Management Information Systems*, 21(4):37–56, Spring 2005.

[FWX04] W. Fan, L. Wang, and W. Xi. Tuning before feedback: combining ranking discovery and blind feedback for robust retrieval. In *ACM SIGIR '04*, pages 138–145, 2004.

[GC97] F. C. Gey and A. Chen. Phrase Discovery for English and Cross-language Retrieval at TREC-6. In *TREC-6, NIST Special Publication*, pages 637–648, 1997.

[GL04] Z. Gu and M. Luo. Comparison of using passages and documents for blind relevance feedback in information retrieval. In *ACM SIGIR'04*, pages 482–483, 2004.

[Har92] D. Harman. Relevance Feedback Revisited. In *ACM SIGIR'92*, pages 77–88, 1992.

[HDO$^+$98] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and Their Applications*, 13(4):18–28, July-August 1998.

[HMIH99] K. Hoashi, K. Matsumoto, N. Inoue, and K. Hashimoto. Query Expansion Method Based on Word Contribution. In *ACM SIGIR'99*, pages 303–304, 1999.

[HTC97] D. Hawking, P. Thistlewaite, and N. Craswell. ANU/ACSys TREC-6 Experiments. In *TREC-6, NIST Special Publication*, pages 275–290, 1997.

[Ide71] E. Ide. *New experiments in relevance feedback. In G. Salton editor, The SMART Retrieval System.* Prentice Hall Inc., 1971.

[IS71] E. Ide and G. Salton. Interactive search strategies and dynamic file organization in information retrieval. In *Rep. ISR-16, Dept. of Computer Science, Cornell Univ., Ithaca, N.Y. Reprinted in the Smart Retrieval System*, 1971.

[Jon72] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. In *Journal of Documentation, 8*, pages 11–21, 1972.

[KGX97] K. L. Kwok, L. Grunfeld, and J. H. Xu. TREC-6 English and Chinese Retrieval Experiments using PIRCS. In *TREC-6, NIST Special Publication*, pages 207–214, 1997.

[KK04] M. S. Khan and S. Khor. Enhanced web document retrieval using automatic query expansion. *Journal of the American Society for Information Science and Technology*, 55(1):29–40, 2004.

[Kwo87] K. L. Kwok. Some consideration for approximate optimal queries. In *ACM Symposium on Applied computing*, pages 19–24, 1987.

[Kwo96] K. L. Kwok. A New Method of Weighting Query Terms for Ad-Hoc Retrieval. In *ACM SIGIR'96*, pages 187–195, 1996.

[LAJ01] A. M. Lam-Adesina and G. J. F. Jones. Applying Summarization Techniques for Term Selection in Relevance Feedback. In *ACM SIGIR'01*, pages 1–9, 2001.

[LPGBA02] C. Lopez-Pujalte, V. P. Guerrero-Bote, and F. M. Anegon. A test of genetic algorithms in relevance feedback. *Information Processing and Management*, 38:793–805, 2002.

[LPGBA03] C. Lopez-Pujalte, V. P. Guerrero-Bote, and F. M. Anegon. Genetic algorithms in relevance feedback: a second test and new contributions. *Information Processing and Management*, 39:669–687, 2003.

[Luh58] H. P. Luhn. The Automatic Creation of Literature Abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.

[MG63] T. Marill and D. M. Green. On the Effectiveness of Receptors in Recognition Systems. *IEEE Transactions on Information Theory*, IT-9:11–17, 1963.

[MO01] H. Mano and Y. Ogawa. Selecting Expansion Terms in Automatic Query Expansion. In *ACM SIGIR'01*, pages 390–391, 2001.

[Moo96] R. J. Mooney. Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In *Empirical Methods in Natural Language Processing*, 1996.

[MR97] M. Magennis and C. J. Van Rijsbergen. The potential and actual effectiveness of interactive query expansion. In *ACM SIGIR'97*, pages 324–332, 1997.

[Ore02] N. Oren. Reexamining tf.idf based information retrieval with Genetic Programming. In *Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology SAICSIT '02*, pages 224–234, 2002.

[Por80] M. F. Porter. An algorithm for suffix strippng. *Program*, 14(3):130–137, 1980.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA., 1993.

[Qui94] J. R. Quinlan. Comparing Connectionist and Symbolic Learning Methods. *Computational Learning Theory and Natural Learning Systems*, 1:445–456, 1994.

[RHP81] C. J. Van Rijsbergen, D. J. Harper, and M. F. Porter. The selection of good search terms. *Information Processing and Management*, 17(2):77–91, 1981.

[RJ76] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[Rob90] S. E. Robertson. On term selection for query expansion. *Journal of Documentation*, 46(4):359–364, 1990.

[Roc71] J. J. Rocchio. *Relevance feedback in information retrieval. In G. Salton editor, The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Inc., 1971.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[Sal71] G. Salton. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Inc., 1971.

[Sal89] G. Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley, 1989.

[SB88] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. 24(5):513–523, 1988.

[SB90] G. Salton and C. Buckley. Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[SBM96] A. Singhal, C. Buckley, and M. Mitra. Pivoted Document Length Normalization. In *ACM SIGIR'96*, pages 21–29, 1996.

[Sin97] A. Singhal. AT&T at TREC-6. In *TREC-6, NIST Special Publication*, pages 215–226, 1997.

[SMT91] J. Shavlik, R. Mooney, and G. Towell. Symbolic and Neural Learning Algorithms: An Experimental Comparison. *Machine Learning*, 6:111–143, 1991.

[Tro04] A. Trotman. An artificial intelligence approach to information retrieval. In *ACM SIGIR'04*, pages 603–607, 2004.

[VB02] E. M. Voorhee and C. Buckley. The effect of topic set size on retrieval experiment errors. In *ACM SIGIR'02*, pages 316–323, 2002.

[VH98] E. M. Voorhees and D. Harman. Overview of the sixth Text REtrieval Conference (TREC-6). In *TREC-6, NIST Special Publication*, 1998.

[VH99] E. M. Voorhees and D. Harman. Overview of the seventh Text REtrieval Conference (TREC-7). In *TREC-6, NIST Special Publication*, 1999.

[VH00] E. M. Voorhees and D. Harman. Overview of the eighth Text REtrieval Conference (TREC-8). In *TREC-8, NIST Special Publication*, 2000.

[Wil96] W. J. Wilbur. Human subjectivity and performance limits in document retrieval. *Information Processing and Management*, 32(5):515–527, 1996.

[WRB$^+$97] S. Walker, S. E. Robertson, M. Boughanem, G. J. F. Jones, and K. Sparck Jones. Okapi at TREC-6 Automatic ad hoc, VLC, routing, filtering and QSDR. In *TREC-6, NIST Special Publication*, pages 125–136, 1997.

[WWL$^+$05] W. S. Wong, H. C. Wu, R. W. P. Luk, H. V. Leong, K. F. Wong, and K. L. Kwok. MATRIX at the TREC2005 Robust Track. In *TREC-2005, NIST Special Publication*, 2005.

[WYB88] S. K. M. Wong, Y. Y. Yao, and P. Bollmann. Linear structure in information retrieval. In *ACM SIGIR'88*, pages 219–232, 1988.

[XC96] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *ACM SIGIR'96*, pages 4–11, 1996.

[YLC76] C. T. Yu, W. S. Luk, and T. Y. Cheung. A statistical model for relevance feedback in information retrieval. *Journal of the ACM*, 23(2):273–286, 1976.

[ZCF$^+$05] B. Zhang, Y. Chen, W. Fan, E. A. Fox, M. Goncalves, M. Cristo, and P. Calado. Intelligent GP fusion from multiple sources for text classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management CIKM '05*, pages 477–484, 2005.