AN AXIOMATIC APPROACH TO INFORMATION RETRIEVAL

BY

HUI FANG

B.S., Tsinghua University, 2001
M.S., University of Illinois at Urbana-Champaign, 2004

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2007

Urbana, Illinois

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0347933. Any opinions, findings and conclusions or recomendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation (NSF).

To Xiaoming and my parents, for their love and support

Acknowledgments

I wish to express my greatest thanks to my advisor and other committee members for their guidance and encouragement, my friends for their help, and my family for their love and tremendous support.

First and foremost, I thank my advisor, ChengXiang Zhai. Cheng introduced me to the exciting field of Information Retrieval, which I enjoy greatly ever since. Cheng is more than just an academic advisor to me. He taught me many things including the importance of aiming high-standards, the art of finding compelling research problems, and the way of doing research. I am inspired by his brilliance, persistence, hard work and dedication. Cheng's nice personality, guidance and encouragement help me go through many obstacles during my graduate study. I could not make this far without him.

I also thank my other thesis committee members, Jiawei Han, Kevin Chang, and Marianne Winslett, for their efforts and constructive suggestions. Their comments are critical in making this thesis more complete, more accurate, and clearer to read. Their suggestions also open my minds to related fields and inspire many possibilities in the future work. I am very fortunate to be a member of Database and Information Systems Laboratory (DAIS), and have the opportunity to work closely with experts in the fields that are related to information retrieval.

I benefited a lot from the discussions with my collaborators. I thank all of them, AnHai Doan, Tao Tao, Qiaozhu Mei, Bin Tan, Xuanhui Wang, Lei Liu, Rishi R. Shiha, Wensheng Wu, Atulya Velivelli, and Azadeh Shakery. I enjoyed a lot when I worked with them.

It has been such a great experience to study in Department of Computer Science at

University of Illinois at Urbana-Champaign. Many other faculty members in the department offered me tremendous help in my study. I am lucky to have them around, and especially thank David Padua, Jennifer Hou, Darko Marinov, AnHai Doan, Geneva Belford, Dan Roth, Stephen Bond, Chandra Chekuri, Saurabh Sinha, Jeff Erickson, and Brian Bailey.

I am lucky to have many good friends and fellow students in UIUC and enjoy every moment that I have spent with them. I wish all of them well in everything: Hong Cheng, Jing Jiang, Deng Cai, Qiaozhu Mei, Tao Tao, Xuehua Shen, Bin Tan, Xin He, Cheng Tao, Xuanhui Wang, Azadeh Shakery, Adam Lee, Alexander Kotov, Bei Yu, William Lee, Shengnan Cong, Bin Yu, Xin Li, Shun Wang, Zheng Shao, Zheng Sun, Xiaoxin Yin, Xinlai Ni, Chao Huang, Jia Guo, Younhee Ko, Xu Ling, Dong Xin, Chengkai Li, Zhen Zhang, Bin He, Seung-won Hwang, Xifeng Yan, Shui-Lung Chuang, Qin Chen, Liang Deng, Hongfei Yan, Lixin Zhou, Chao Liu, Rishi R. Sinha, Qingbo Zhu, Mingliang Wei, Changhao Jiang, Xiaohui Gu, Zhen Wen, Zhenmin Li, Chengdu Huang, and Liqian Luo.

Finally, I would like to thank my husband Xiaoming for his love and strong support for my research. My acknowledgements can not be finished without saying thanks to my parents and my sister, for their unwavering support throughout everything. This thesis is dedicated to them.

Abstract

With the birth of Web, the amount of information grows rapidly. Such a huge amount of information poses significant challenges in text information management. Search engines are by far the most powerful tools that help users find information. The accuracy of search engines significantly affects our productivity and our quality of life. Text retrieval is the underlying research problem behind all the search engines. An improved test retrieval model enables every search engine to achieve higher search accuracy.

The thesis presents a novel axiomatic framework to study and develop more robust and effective text retrieval models. The current retrieval models all model relevance *indirectly*, which prevents us from understanding what makes a retrieval function perform well. As a result, we have to rely on heavy parameter tuning to optimize the retrieval performance. To overcome this limitation, the proposed axiomatic framework models the relevance *directly* with a set of retrieval constraints (i.e., axioms). Our approach is motivated by the empirical observation that good retrieval performance is closely related to the use of various retrieval heuristics. We formalize these retrieval heuristics as constraints, and use them as guidance on diagnosing the weaknesses and strengths of a retrieval function and developing more robust and effective retrieval functions in a principled way. Experiments show three major benefits of the proposed axiomatic approach. First, it allows us to diagnose the weaknesses and strengths of retrieval functions both analytically and empirically. The performance of retrieval functions can be improved based on the diagnostic results. Second, the axiomatic approach makes it possible to derive more robust and effective retrieval functions. The derived new retrieval functions are more robust and less sensitive to parameter settings than the existing retrieval functions with comparable optimal performance. Third, the axiomatic approach provides an easy way to incorporate additional information, such as semantic term matching, to further improve the retrieval performance.

The axiomatic framework opens up many promising new directions for studying and developing more robust and effective retrieval functions. Since relevance is directly modeled through retrieval constraints, the framework enables us to understand relevance theoretically and predict the performance of a retrieval function analytically. The framework facilitates diagnostic analysis of retrieval functions, thus can provide guidances on how to eventually develop the ultimate optimal retrieval function.

Table of Contents

Chapter 1

Introduction

Information retrieval is a field that concerns with helping users find useful information from large document collections. The field becomes more important after the birth of web, because the amount of online information grows explosively. Various kinds of text information, such as web pages, emails, instant messages, literature and blogs, are being continuously produced everywhere in the world in every possible way. Recent study [31] shows that there were about 5 exabytes of new information in 2002. Specifically, the World Wide Web contained about 170 terabytes of information on its surface, instant messaging generated about 274 terabytes of information a year, and emails generated about 400,000 terabytes of new information each year worldwide. There is no doubt that the information will keep growing in the following years. Unfortunately, the growth makes it more challenging to manage useful information effectively and efficiently.

Users are overwhelmed with the huge amount of information and have an urgent need for more powerful information retrieval systems. Search engines are by far the most powerful tools that help users find information, and have become more and more essential in all aspects of our life. It is estimated that 6 billion user queries were submitted to search engines in October, 2006 alone. It is clear that the accuracy of search engines significantly affects our productivity and our quality of life.

Text retrieval is the underlying research problem behind all the search engines. An improved text retrieval model enables every search engine to achieve higher search accuracy, because textual information is the most common type of information and can be used to describe other types of information. Text retrieval problem is usually defined as identifying all the documents satisfying a user's information need from a collection. A user expresses the information need as a query. If a document satisfies a user's information need, it is relevant to the corresponding query. Due to the inherent vagueness of relevance, it is hard to find a clear boundary between relevant documents and non-relevant documents. Moreover, even if two documents are relevant, one of them might be more relevant than the other. Therefore, a retrieval system usually assigns a relevance score to every document in the collection and returns a ranking list of the documents based on the relevance scores [45]. The key challenge of the text retrieval problem is how to model *relevance* appropriately. Specifically, the challenge is how to derive a retrieval function to compute the relevance score of each document for the given query.

It has always been a significant challenge to develop principled text retrieval models that are effective, robust, and efficient. Although many text retrieval models have been proposed and studied [63, 52, 50, 48, 42, 65, 64, 14, 38, 26, 27], they generally do not offer any guarantee for optimal retrieval performance. Non-optimal parameter setting easily causes a model to perform poorly. As a result, heavy parameter tuning is almost always necessary to achieve optimal performance on a particular data set. In a way, this limitation is caused by the way existing retrieval models are developed. Most existing models have been developed based on a "coarse" approximation of the notion of relevance at the level of documents and queries, which often converts a retrieval problem to some other problem in a framework without relevance. For example, in vector space models, the notion of relevance is assumed to be captured through a similarity measure between a query vector and a document vector, which allows us to conveniently convert the retrieval problem to one mainly involving vector space operations [50]. Similarly, in probabilistic retrieval models, including language modeling approaches, the notion of relevance is assumed to be captured through a binary random relevance variable and a probabilistic model is defined to associate this variable with some probabilistic representation of documents and queries, which again allows us to avoid directly addressing the notion of relevance and to conveniently convert the retrieval problem to one involving defining and estimating probabilistic models [26]. The lack of a detailed modeling of relevance makes it impossible to predict a model's empirical performance and to provide guidance on how to develop a retrieval model with optimal performance. Thus, heuristic modification of a retrieval formula and heuristic introduction of additional parameters are often made to improve retrieval performance. To avoid such heuristic modifications, we will need to capture relevance *directly* at a finer granularity level of terms.

In this thesis, we propose a novel general axiomatic framework to study and develop retrieval models. In the axiomatic framework, we study how a retrieval function should behave, formalize necessary properties of reasonable retrieval functions as retrieval constraints, and model the relevance with these formalized constraints. The essential difference of our approach from the existing work is that we model relevance more directly through the term-level retrieval constraints.

The axiomatic approach is motivated by the observation that good retrieval performance is closely related to the use of various retrieval heuristics, especially TF-IDF weighting and document length normalization. Many empirically effective retrieval formulas tend to boil down to an explicit or implicit implementation of these retrieval heuristics, even though they may be motivated quite differently [68]. Thus, we formally define a set of basic desirable constraints that any reasonable retrieval function should satisfy. They capture the commonly used retrieval heuristics, such as TF-IDF weighting, in a formal way, making it possible to apply them to any retrieval formula analytically. We show that satisfaction of retrieval constraints is closely related to empirical performance of a retrieval function. These retrieval constraints serve as the foundation of the framework, and make it possible to address several limitations of existing retrieval functions and evaluation methodology.

Existing evaluation methodology provides little explanation for the performance differences among retrieval functions. The axiomatic approach has the unique advantage to overcome this limitation, because direct relevant modeling ties the constraint satisfaction with empirical retrieval performance, which makes it possible to diagnose the weaknesses and strengths of a retrieval function through checking how well a retrieval function implement retrieval constraints. Although many retrieval functions implement the retrieval constraints, their implementations are different and it is unclear at all whether one implementation of a heuristic is better or worse than the others. Ideally, if we can identify the problematic implementations of the heuristics in a retrieval function (i.e., the weaknesses of a retrieval function), the retrieval performance could be further improved. To achieve this goal, we propose two strategies to examine how well a retrieval function implements the retrieval heuristics. The first strategy is to use the formalized retrieval constraints to *analytically* check the implementations in retrieval functions. The second strategy is to define a set of relevance-preserving perturbations and perform diagnostic tests to *empirically* evaluate how well a retrieval function implement retrieval heuristics. Experiments show that both strategies are effective to identify the potential problems of the implementations of retrieval heuristics. The empirical performance of retrieval functions can be improved after we fix these problems.

The axiomatic framework allows us not only to diagnose the weaknesses of existing retrieval functions, but also to derive more robust and effective retrieval functions through *direct* relevance modeling. The basic idea is to search for a retrieval function that satisfies a set of retrieval constraints that any reasonable retrieval function should satisfy. In order to efficiently search for a retrieval function, we propose to define the function space inductively. In particular, based on the inductive definition, a retrieval function can be decomposed into three component functions, referred to as *Primitive weighting function*, *Query growth function* and *Document growth function*, respectively. Thus searching for a good retrieval function boils down to searching for a good formula for each of these three functions in our constrained search space. We then use the formalized retrieval constraints and the technique of exploratory data analysis [18, 20] to constrain the choices for the three component functions and derive several new retrieval functions. We implement and test these new functions with a number of representative test sets. The experiment results show that the derived new functions are more robust and less sensitive to parameter settings than the existing retrieval functions with comparable optimal performance.

The framework also provides a natural way to incorporate additional useful information to further improve the performance. In particular, we study how to incorporate semantic term matching in the thesis. Most traditional retrieval models compute relevance scores solely based on *exact* (i.e., syntactic) matching of terms in the query and documents, without allowing distinct but semantically related terms to match each other and contribute to the retrieval score. The syntactic term matching is clearly a limitation that hinders retrieval performance, because it is unlikely that the authors of relevant documents always use exactly the same terms as a user would use in a query. For example, given the query "car", intuitively, a single-term document with the term "vehicle" should have a higher score than a single-term document with the term "fish" because "car" is semantically more related to "vehicle" than "fish". To address this limitation, we formally define several constraints on semantic term matching, and then use these constraints to provide guidance on how to compute the term semantic similarity and how to regularize the weights of the original terms and the semantically related terms. We show that the proposed semantic expansion works well and significantly improves the retrieval accuracy over the original baseline axiomatic retrieval functions on all the data sets we experimented with. Moreover, the analysis of semantic term matching constraints can predict parameter boundaries that are consistent with the empirically discovered optimal ranges of parameters. Furthermore, as a pseudo feedback method, our method outperforms a state-of-the-art language modeling approach for pseudo feedback [76] due to its capability of selecting better terms for query expansion.

The axiomatic approach provides a promising new framework to study retrieval functions. The framework makes it possible to predict the empirical performance of a retrieval function analytically, to diagnose the weaknesses and strengths of retrieval functions, and to develop more robust and effective retrieval functions. The axiomatic framework opens up many new directions to study retrieval function, and it has great potential to allow us to develop the optimal retrieval functions, and to achieve theoretical understanding of relevance.

The rest of the thesis is organized as follows. First, we discuss related work in Chapter 2. We describe the basic idea of axiomatic approach in Chapter 3. We discuss how to diagnose the weaknesses and strengths of a retrieval function analytically in Chapter 4, and empirically in Chapter 5. We explain how to derive a new retrieval function with the axiomatic approach in Chapter 6, and then present how to incorporate semantic term matching to further improve the performance in Chapter 7. Finally, we summarize the contributions of the thesis and discuss future work in Chapter 8.

Chapter 2

Related Work

The axiomatic approach provides a novel framework to study and develop retrieval models. In the thesis, we focus on solving the following three research questions: (1) How to diagnose the weaknesses of a retrieval function in order to modify the function to improve the performance? (2) How to develop more robust and effective retrieval function by modeling relevance *directly* with term-level retrieval constraints? (3) How to incorporate semantic term matching in retrieval functions to further improve the search accuracy? A tremendous amount of effort has been devoted to solving these research questions. We now discuss the existing work in these three aspects and the limitations in details.

2.1 Diagnostic Evaluation

Many large evaluation collections have been constructed since the beginning of the TREC [68]. An evaluation collection includes a document collection, a set of queries and a set of relevance judgements indicating which documents are relevant to which queries. TREC conferences employ system pooling to obtain a set of relevance judgements. The system pooling method has been shown to be sufficient for search purposes [70]. Most studies on evaluation of retrieval models focus on how to create a better pool and how to reduce the pool size while maintain the confidence of the evaluation results [10, 78, 60, 53, 8, 9]. Such an evaluation methodology is not informative enough to directly explain performance differences among retrieval functions or provide guidance on how to improve the performance of a retrieval function.

Studies in pivoted normalization function [57, 58] demonstrate that a retrieval function can be improved if we could pinpoint the weakness of a retrieval function. Singhal et. al. [57] observed the deficiency of a particular length normalization method, and proposed pivoted normalization technique to modify the normalization function. They later noticed the poor performance of the logarithmic TF-factor on a TREC collection, found another deficiency of the implementation of term frequency (TF)part [58], and modified the TF part accordingly. However, their methods are not general enough to be easily applied to identify weaknesses in other aspects.

Recent studies in RIA workshop [19] focus on understanding the empirical behavior of retrieval functions. But they did not focus on identifying the weaknesses and strengths of retrieval functions.

Unlike all the existing work, we propose a general evaluation methodology to empirically and analytically *diagnose* the weaknesses and strengths of retrieval functions. Such a methodology can provide explanation for the empirical differences among retrieval functions and pinpoint the places where we should modify to further improve the performance of a retrieval function.

2.2 Retrieval Models

A large number of different retrieval models have been proposed and studies. Although these models are motivated quite differently, the derived retrieval functions are similar in the sense that they all model relevance *indirectly* and all consider the similar characteristics of a term. Before we review the related work on retrieval models, we first explain some notations which will be used in the thesis.

$S(Q, D)$ is the relevance score of a document D for the given query Q.

$c(t, D)$ is the count of word t in the document D.

$c(t, Q)$ is the count of word t in the query Q.

N is the total number of documents in the collection.

$df(t)$ is the number of documents that contain the term t.

$|D|$ is the length of document D.

$avdl$ is the average document length.

$|Q|$ is the length of query Q.

$p(t|C)$ is the probability of a term t given by the collection language model [77].

Three representative retrieval models are vector space models [52, 50, 48], probabilistic models [42, 65, 64, 14, 38, 26], and inference-based models [67, 74, 13]. In this section, we briefly review these major approaches.

Vector space models [50, 51] represent any text, including documents and queries, as a vector in a high-dimensional space, where each term defines an independent dimension. The value of each vector component is related to the weight of corresponding term in the given text. Unfortunately, the vector space models do not provide any guidance on how to define term weights. Term weighting is usually assigned based on different heuristics, such as TF-IDF weighting and document length normalization. In the vector space models, the notion of relevance is captured through a similarity measure on a query vector and a document vector. The relevance score between a query and a document can be computed based on the similarity between the two corresponding vectors. The two most commonly used similarity functions are cosine similarity and dot product. In this way, the retrieval problem is converted to one involving vector space operations [50]. The term weighting scheme is totally outside the relevance modeling framework, which makes it hard to control and explain parameters in the retrieval models. As a result, retrieval performance is often sensitive to parameter setting. One of the best performing vector space retrieval function is pivoted normalization retrieval function [57, 56], which is shown as follows.

$$S(Q,D) = \sum_{t \in D \cap Q} \frac{1 + ln(1 + ln(c(t,D)))}{(1-s) + s\frac{|D|}{avdl}} \cdot c(t,Q) \cdot ln\frac{N+1}{df(t)}$$

Probabilistic retrieval models [42, 65, 64, 14, 38, 26, 62] model the relevance by estimating the probability that a document is relevant to a given query. In the probabilistic models, the notion of relevance is captured through the binary random relevance variable and a probabilistic model is defined to associate this variable with some (probabilistic) representation of documents and queries. In this way, the retrieval problem is converted to the problem that involves defining and estimating probabilistic models [26]. Similar to the problems in the vector space models, the models can not provide clear guidelines on model estimation and parameter setting. Thus, although the high level formulation is elegant, there exists a significant gap between this high level formulation and the real retrieval formulas. Such a gap is often filled by different heuristics.

Okapi [43] is a highly effectiveness retrieval function that represents the classical probabilistic model. The function as presented in [56] is [1]

$$S(Q,D) = \sum_{t \in Q \cap D} \left(ln\frac{N - df(t) + 0.5}{df(t) + 0.5} \times \frac{(k_1 + 1) \times c(t,D)}{k_1((1-b) + b\frac{|D|}{avdl}) + c(t,D)} \times \frac{(k_3 + 1) \times c(t,Q)}{k_3 + c(t,Q)} \right)$$

where k_1 (between 1.0-2.0), b (usually 0.75), and k_3 (between 0-1000) are constants.

Dirichlet prior is one of the best performing retrieval functions that are derived using language modeling approaches [77]. This function uses the Dirichlet prior smoothing method to smooth a document language model and then ranks documents according to the likelihood of the query according to the estimated language model of each document. With a notation

[1]There is a typo in the formula in [56], which is corrected here.

consistent with the formulas above, the Dirichlet prior retrieval function is

$$S(Q,D) = \sum_{t \in Q \cap D} c(t,Q) \cdot ln(1 + \frac{c(t,D)}{\mu \cdot p(t|C)}) + |Q| \cdot ln \frac{\mu}{|D| + \mu}$$

$p(t|C)$ is similar to the document frequency $df(t)$, and it indicates how popular the term t is in the whole collection.

Inference-based models [67, 74, 13, 64, 15, 40, 41, 25] model the relevance of a document with respect to a query by the uncertainty associated with inferring the query from the document. Various definitions of "inferring a query from a document," lead to different inference models. While theoretically interesting, the probabilistic inference models must rely on further assumptions about the representation of documents and queries in order to obtain an operational retrieval formula. The choice of such representations is in a way outside the model, so there is little guidance on how to choose or how to improve a representation.

Summary: As explained, most existing models have been developed based on a "coarse" approximation of the notion of relevance at the level of documents and queries, which often converts a retrieval problem to some other problem in a framework without relevance. The lack of a detailed modeling of relevance makes it difficult for the models to achieve optimal retrieval performance. Thus, heuristic modification of a retrieval function and heuristic introduction of additional parameters are often made to improve retrieval performance.

Unlike the previous work, we propose an axiomatic approach to the retrieval problem. We formally define retrieval heuristics as constraints on retrieval functions and develop a novel axiomatic framework to capture the relevance directly at a finer granularity level of terms. Such a framework allows us to specify the properties that we would like a retrieval function to satisfy. These formal properties, i.e., constraints, can provide guarantee of good retrieval performance for any derived retrieval function. Axiomatic approaches have been studied previously in information retrieval, mostly based on logic [6, 21, 73], but, as far as we know, none of these studies has resulted in any effective retrieval formula. Although the general idea is similar, our approach is completely different from these previous studies both in the space of retrieval functions considered and in the way we specify the axioms (i.e., constraints).

2.3 Semantic Term Matching

Many existing retrieval functions assume that relevance score can be computed solely based on the exact matching of query terms. In reality, the assumption does not hold, because it is unlikely that a user would use a query term that is exactly the same one as used in relevant documents. Many studies have tried to bridge the vocabulary gap between documents and queries in traditional retrieval models, mostly based on either co-occurrence-based thesaurus [28, 59, 37, 39, 23, 75, 54, 4] or hand-crafted thesaurus [69, 30]. Some researchers used both [33, 7]. Although the axiomatic approach is applicable to exploit both types of thesauri, in this thesis, we focus on the use of co-occurrence-based thesaurus and leave other possibilities as future work.

The earliest study of co-occurrence-based thesaurus can be traced back to the early sixties. Lesk [28] studied term expansion in the vector space model, where term similarity is computed based on the cosine coefficient [50]. Smeaton et al. [59] studied query expansion based on classical probabilistic model. These previous studies suggested that query expansion based on term co-occurrences is unlikely to significantly improve performance [37]. Qiu et al. [39] showed that adding terms that have the greatest similarity to the entire query, rather than individual terms, can obtain more improvement. Xu et al. [75] showed that the analysis of word occurrences and relationships on a local set of documents (i.e. the top ranked documents retrieved by the original query) yields better performance than on the whole corpus. In language modeling approaches, Berger et al. [5] proposed a translation model to incorporate term relationship into language modeling approaches. Cao et al. [7] extended the translation model to integrate both co-occurrence and hand-crafted thesaurus and achieve reasonable performance improvement. Bai et al. [4] showed that query expansion based on co-occurrences can improve the performance in language modeling approaches.

In the thesis, we take the axiomatic approach and use constraints to guide us in the incorporation of semantic term matching, which is the major difference between our approach and existing work. We integrate term semantic relationship as a component in the retrieval models, and show that this method can be implemented as query expansion, which is similar to previous work [28, 59, 37, 39, 54, 4]. Furthermore, our approach bears some similarity to traditional feedback methods [47, 76, 38, 35], which also select terms from documents as to expand the query. But our method selects terms that are semantically related to each individual query term and relies on the axiomatic approaches to combine them, while feedback methods select terms that discriminate the feedback documents, which are not necessarily related to any individual query term. Because of this difference, our method is complementary to the traditional feedback method. Indeed, our experiment results show that they can be combined to further improve performance.

Chapter 3

Our Approach - An Axiomatic Framework for IR

This dissertation focuses on a novel axiomatic approach to information retrieval. The axiomatic approach is inspired from the empirical observation that good retrieval performance is closely related to the use of various retrieval heuristics, especially TF-IDF weighting and document length normalization. Many empirically effective retrieval functions tend to boil down to an explicit or implicit implementation of these retrieval heuristics, even though they may be motivated quite differently [68]. Even the recently developed language modeling approach has been shown to be connected with these heuristics [77]. It thus appears that these heuristics are somehow necessary for achieving good retrieval performance. However, it is unclear at all what exactly are these "necessary heuristics" mathematically. Thus, the two basic research questions are then how we can *formally* define and characterize these necessary retrieval heuristics (i.e., constraints), and how we can use these retrieval constraints to diagnose weaknesses and develop effective retrieval functions.

To address these research questions, we first start from the desirable properties of relevance and formalize them as retrieval constraints, i.e., axioms. We then use these constraints as guidance in identifying the weaknesses of a retrieval function and searching for an effective function in a space of all possible candidate retrieval functions.

3.1 Constraints for Retrieval Functions

The proposed axiomatic approach is motivated by the following observations on some common characteristics of typical retrieval formulas. First, most retrieval methods assume a "bag of words" (more precisely, "bag of terms") representation of both documents and queries. Second, a highly effective retrieval function typically involves a TF part, an IDF part, and a document length normalization part [49, 79]. The TF part intends to give a higher score to a document that has more occurrences of a query term, while the IDF part is to penalize words that are popular in the whole collection. The document length normalization is to avoid favoring long documents; long documents generally have more chances to match a query term simply because they contain more words. Finally, different retrieval formulas do differ in their way of combining all these factors, even though their empirical performances may be similar. These observations suggest that there are some "basic requirements" that all reasonable retrieval formulas should follow. For example, if a retrieval formula does not penalize common words, then it somehow violates the "IDF requirement", thus can be regarded as "unreasonable." However, some of these requirements may compromise each other. For example, while the TF heuristic intends to assign a higher score to a document that has more occurrences of a query term, the document length normalization component may cause a long document with a higher TF to receive a lower score than a short document with a lower TF. Similarly, if two documents match precisely one single, but different query term, the IDF heuristic may allow a document with a lower TF to "beat" the one with a much higher TF. A critical question is thus how we can regulate such interactions so that they will all be "playing a fair game"? Clearly, in order to answer this question, we must first define what is a "fair game", i.e., we must define what exactly is a *reasonable* retrieval function. In this dissertation work, we formally define a set of retrieval constraints by formalizing the above intuitive retrieval heuristics, and show that the satisfaction of these constraints is closely related to the empirical performance of a retrieval function.

3.1.1 Formal Definitions of Basic Retrieval Constraints

In this subsection, we formally define seven intuitive and desirable constraints that any reasonable retrieval formula should satisfy. They capture the commonly used retrieval heuristics, such as TF-IDF weighting, in a formal way, making it possible to apply them to any retrieval formula analytically.

These most commonly used retrieval heuristics are summarized as follows.

- A highly effective retrieval function typically involves a TF part, an IDF part, and a document length normalization part [49, 79].

- The TF part intends to give a higher score to a document that has more occurrences of a query term.

- The IDF part is to penalize words that are popular in the whole collection.

- The document length normalization is to avoid favoring long documents since long documents generally have more chances to match a query term simply because they contain more words.

Note that these heuristics are *necessary*, but not necessarily sufficient, and should not be regarded as the *only* constraints that we want a retrieval function to satisfy; indeed, it is not hard to come up with additional constraints that may also be reasonable. However, we focus on only seven basic constraints in this thesis because they capture the major well-known IR heuristics, particularly TF-IDF weighting and length normalization. As shown in Chapter 4, the empirical performance of a retrieval function is tightly related to how well it satisfies these constraints. Thus, the proposed constraints provide a good explanation of many empirical observations about retrieval functions.

Term Frequency Constraints (TFCs)

We define three constraints to capture the desired scoring of a term frequency.

TFC1: Let Q be a query and D be a document. If $q \in Q$ and $t \notin Q$, we must have $S(Q, D \cup \{q\}) > S(Q, D \cup \{t\})$.

TFC2: Let Q be a query and D be a document. If $q \in Q$ and $t \notin Q$, $S(Q, D \cup \{q\} \cup \{t\}) - S(Q, D \cup \{t\} \cup \{t\}) > S(Q, D \cup \{q\} \cup \{q\}) - S(Q, D \cup \{q\} \cup \{t\})$.

TFC3: Let D be a document and $Q = \{q_1, q_2\}$ be a query with two query terms q_1 and q_2, where $td(q_1) = td(q_2)$ and $td(t)$ can be any reasonable measure of term discrimination value. If $q_1 \in D$ and $q_2 \notin D$, $S(Q, D \cup \{q_1\}) < S(Q, D \cup \{q_2\})$.

The first constraint captures the basic TF heuristic, which gives a higher score to a document with more occurrences of a query term when the only difference between two documents is the occurrences of the query term. In other words, the score of retrieval formula will increase with the increase in TF (i.e., the first partial derivative of the formula w.r.t. the TF variable should be positive). The second constraint ensures that the increase in the score due to an increase in TF is smaller for larger TFs (i.e., the second partial derivative w.r.t. the TF variable should be negative). Here, the intuition is that the change in the score caused by increasing TF from 1 to 2 should be larger than that caused by increasing TF from 100 to 101. The third constraint implies another desirable property - if two documents have the same total occurrences of all query terms and all the query terms have same term discrimination value, a higher score will be given to the document covering more distinct query terms.

Term Discrimination Constraint (TDC)

We define this constraint to capture the desired term discrimination scoring.

TDC: Let D be a document and $Q = \{q_1, q_2\}$ be a query. Assume there are two documents D_1 and D_2, where $|D_1| = |D_2|$, D_1 contains only q_1 and D_2 contains only q_2. If $td(q_1) > td(q_2)$, $S(Q, D \cup D_1) > S(Q, D \cup D_2)$.

$td(t)$ can be any reasonable measure of term discrimination value (usually based on term popularity in a collection). It gives higher weights to more discriminative terms. Any IDF-like function can be a good candidate for td. This constraint implies that we need to penalize the terms popular in the collection. This constraint is essentially the M-TDC (modified TDC) [55].

Based on TFC2 and TDC, the following constraint can be derived[1].

Let D be a document and Q be a query. If $q_1 \in Q$, $q_2 \in Q$, $q_1 \in D$, $q_2 \in D$, $td(q_1) > td(q_2)$ and $c(q_1, D) \leq c(q_2, D)$, $S(Q, D \cup \{q_1\}) > S(Q, D \cup \{q_2\})$.

Length Normalization Constraints (LNCs)

We define two constraints to quantify the penalty on long documents.

LNC1: Let Q be a query and D be a document. If for some word $t \notin Q$, then $S(Q, D) \geq S(Q, D \cup \{t\})$.

LNC2: Let Q be a query and D be a document. If $D \cap Q \neq \emptyset$ and D' is formed by concatenate D with itself k times, then $S(Q, D') \geq S(Q, D)$.

The first constraint says that the score of a document should decrease if we add an extra occurrence of a "non-relevant word" (i.e., a word not in the query). The second constraint intends to avoid over-penalizing long relevant documents, as it says that if a document D has at least one query term, and we concatenate the document with itself k times to form a new document, then the relevance score of the new document should not be lower than the original one. Here, we make the assumption that the redundancy issue is not considered.

TF-LENGTH Constraint (TF-LNC)

We define this constraint to avoid over-penalizing long documents.

[1]This constraint is the relaxed-formulation of original TDC which was defined in [11]. We drop the original one since it is too strong in some sense.

TF-LNC: Let Q be a query and D be a document. If for some word $q \in Q$, then $S(Q, D) \leq S(Q, D \cup \{q\})$.

It ensures that the relevance score would not decrease after adding more query terms to a document.

Table 3.1: Summary of intuitions for each formalized constraint

Constraints	**Intuitions**
TFC1	to favor a document with more occurrence of a query term
TFC2	to ensure that the amount of increased score due to adding a query term must decrease as more terms are added
TFC3	to favor a document matching more distinct query terms
TDC	to penalize the words popular in the collection
TDC	to assign higher weights to discriminative terms
LNC1	to penalize a long document(assuming equal TF)
LNC2, TF-LNC	to avoid over-penalizing a long document
TF-LNC	to regulate the interaction of TF and document length

Table 3.1 summarizes the intuitions behind each formalized constraint. In fact, TFC1 can be derived from LNC1 and TF-LNC. We still present TFC1 in the thesis, because it is the most intuitive constraint. All the other defined constraints are basic and non-redundant in the sense that none of them can be derived from the others. Formally, suppose C_i represents the set of all the retrieval functions satisfying the i-th Constraint, it can be shown that $\forall i, j$, where i-th Constraint and j-th Constraint are not redundant, $\exists S' \in C_i$, such that $S' \in C_i - C_j$. For example, $S'(Q, D) = \sum_{t \in D} c(t, D)$. It satisfies TF-LNC but fails to satisfy TDC, LNC1, TFC2 and TFC3.

We must emphasize once again that the constraints proposed in this section are necessary for a reasonable retrieval formula, but not necessarily sufficient, and should not be regarded as the *only* constraints that a reasonable retrieval formula has to satisfy. When any constraint is violated, we know the retrieval function may not perform well empirically, but satisfying all the constraints does not necessarily guarantee good performance. Ideally, we want as many constraints as possible so that we can effectively prune the search space and find an effective function more easily. In reality, however, when adding more constraints, we will inevitably introduce bias. Additionally, some constraints may be too strong or even contradictory. The key challenge is to find more retrieval constraints without introducing unreasonable bias.

3.2 Diagnostic Evaluation

A retrieval function is typically evaluated using standard test collections and evaluation measures such as the Mean Average Precision (MAP) and precision at 10 documents, which generally reflect the utility of a retrieval function. Unfortunately, such an evaluation methodology provides little *explanation* for the performance differences among retrieval functions. For example, comparing two retrieval functions based on MAP, we can only know which function gives an overall better ranking of documents on a particular data set, but it is hard to identify the underlying causes of such performance difference. Since being able to identify the weaknesses of a retrieval function is necessary for further improving the retrieval function [57, 58], a very interesting research question is how to design a new evaluation methodology to help identify the strengths and weaknesses of retrieval functions.

In the axiomatic framework, the formalized retrieval constraints tie retrieval performance closely with constraint satisfaction. Thus, we propose two strategies to diagnose the weaknesses and strengths of a retrieval function through checking how well it implements the constraints. The first strategy is to use the formalized constraints to *analytically* check the implementations in retrieval functions. The second strategy is to define a set of relevance-preserving perturbations and perform diagnostic tests to *empirically* evaluate how well a retrieval function implement retrieval heuristics.

These two strategies are both important and complementary to each other. Constraint analysis allows us to predict the empirical performance of a retrieval function analytically, while perturbation-based diagnostic tests make it possible to explain the performance difference of two retrieval functions even if they have the same constraints analysis results.As shown in Chapter 4 and 5, both strategies are effective to identify the potential problems in implementations of the retrieval heuristics. The performance of retrieval functions can be improved after we fix these problems.

3.3 Derivation of Retrieval Functions

Despite the progress in the development of formal retrieval models, good empirical performance rarely comes directly from a theoretically well-motivated model; rather, heuristic modification of a model is often necessary in order to achieve optimal retrieval performance. In some way, this limitation is caused by the indirect relevance modeling of the existing models. To overcome this limitation, we propose to model relevance directly with the axiomatic approach.

The basic idea is to search in a space of candidate retrieval functions for one that can satisfy reasonable retrieval constraints. Our assumption is that if a retrieval function satisfies all the desirable retrieval properties, it would likely be effective empirically. Compared with other retrieval models, the axiomatic approach has the advantage of capturing relevance more directly at the term level with formalized retrieval constraints. In order to implement this idea, we have to address two challenges: how to define a search space of all possible retrieval functions and how to search efficiently in the space.

A reasonable function space must be large enough to include all effective retrieval functions, yet small enough for efficient search. Following most existing retrieval models, we assume that both documents and queries are "bags of terms", i.e. we ignore the order of terms in documents or queries. Such simplification has been working well empirically, and efforts trying to break such limitation have so far not been very successful. To make our framework as general as possible, we include all the real-valued scoring functions defined on a bag-of-term representation of documents and queries. Formally, let T be the set of all terms. Let query $Q = \{q_1, ..., q_n\}$ and document $D = \{d_1, ..., d_m\}$ be two bags of terms, where $q_i, d_i \in T$, and it is possible that $q_i = q_j$ or $d_i = d_j$ even if $i \neq j$. A scoring function S can be defined as follows:

$$S : Q \times D \rightarrow \Re \tag{3.1}$$

With the defined function space and desirable retrieval constraints, our goal is to search for a function that satisfies all the retrieval constraints in the function space. The challenge is how to search for such function efficiently. One possible solution is to only focus on part of the whole function space, e.g., the inductive definition as we will explain in Chapter 6.

The proposed axiomatic approach also provides a natural way to incorporate more information to further improve the retrieval performance. To implement such an idea, we need to follow two steps. We should first define a set of retrieval constraints for these useful information, and then extend the retrieval functions to make them satisfy the additional set of constraints. In the thesis, we study how to incorporate semantic term matching to further improve the performance in Chapter 7.

Chapter 4

Analytical Diagnostic Evaluation

In this chapter, we check the formalized constraints, which are defined in the previous chapter, on a variety of retrieval formulas, which respectively represent the vector space model (pivoted normalization), the classic probabilistic retrieval model (Okapi), and the recently proposed language modeling approach (Dirichlet prior smoothing). We find that none of these retrieval formulas satisfies all the constraints unconditionally, though some formulas violate more constraints or violate some constraints more "seriously" than others. Empirical results show that when a constraint is not satisfied, it often indicates non-optimality of the method, and when a constraint is satisfied only for a certain range of parameter values, its performance tends to be poor when the parameter is out of the range. In general, we find that the empirical performance of a retrieval formula is tightly related to how well it satisfies these constraints. Thus the proposed constraints provide a good explanation of many empirical observations about retrieval methods. Moreover, these constraints make it possible to evaluate any existing or new retrieval formula *analytically* and suggest how we may further improve a retrieval formula.

4.1 Constraint Analysis

In this section, we apply the seven constraints defined in the previous chapter to three specific retrieval formulas, which respectively represent the vector space model (pivoted normalization), the classic probabilistic retrieval model (Okapi), and the recently proposed language modeling approach (Dirichlet prior smoothing). Our goal is to see how well each retrieval formula satisfies the proposed constraints and how closely the constraint analysis results are related to the empirical performance of a retrieval function. As will be shown, it turns out that none of these retrieval formulas satisfies all the constraints *unconditionally*, though some models violate more constraints or violate some constraints more "seriously" than others. The analysis thus suggests some hypotheses regarding the empirical behavior of these retrieval formulas. Furthermore, empirical results show that when a constraint is not satisfied, it often indicates non-optimality of the method, and when a constraint is satisfied only for a certain range of parameter values, its performance tends to be poor when the parameter is out of the range. In general, we find that the empirical performance of a retrieval formula is tightly related to how well it satisfies these constraints. Thus the proposed constraints provide a good explanation of many empirical observations about retrieval methods. Moreover, these constraints make it possible to evaluate any existing or new retrieval formula *analytically* and suggest how we may further improve a retrieval formula.

4.1.1 Pivoted Normalization

Pivoted normalization is one of the best performing vector space retrieval function is pivoted normalization retrieval function [57, 56], which is shown as follows.

$$S(Q,D) = \sum_{t \in D \cap Q} \frac{1 + ln(1 + ln(c(t,D)))}{(1-s) + s\frac{|D|}{avdl}} \cdot c(t,Q) \cdot ln\frac{N+1}{df(t)}$$

Table 4.1: Constraint analysis results (Pivoted)

TFCs	TDC	LNC1	LNC2	TF-LNC
Yes	Yes	Yes	Cond.	Cond.

The results of analyzing the pivoted normalization formula are summarized in Table 4.1, where TFCs means TFC1, TFC2, and TFC3. It is easy to prove that TFCs, TDC and LNC1 can be satisfied unconditionally. We now examine some of the non-trivial constraints.

First, let us examine the TF-LNC constraint. Consider a common case when $|D_1| = avdl$. It can be shown that the TF-LNC constraint is equivalent to the following constraint on the parameter s:

$$s \leq \frac{l(c(t, D_1)) - l(c(t, D_2))}{(c(t, D_1) - c(t, D_2)) \times (1 + l(c(t, D_1)))} \times avdl$$

where $l(x) = ln(1 + ln(x))$.

This means that TF-LNC is satisfied only if s is below a certain upper bound. The TF-LNC constraint thus provides an upper bound for s, which is tighter for a larger $c(q, D_1)$. However, when $c(t, d_1)$ is small, TF-LNC constraint does not provide any effective bound for s, since $s \leq 1$.

Finally, we show that the LNC2 leads to an upper bound for parameter s as well. The LNC2 constraint is equivalent to

$$\frac{1 + ln(1 + ln(k \times c(t, D_2)))}{1 - s + s\frac{k \times |D_2|}{avdl}} \cdot c(t, Q) \cdot ln\frac{N+1}{df(t)} \geq \frac{1 + ln(1 + ln(c(t, D_2)))}{1 - s + s\frac{|D_2|}{avdl}} \cdot c(t, Q) \cdot ln\frac{N+1}{df(t)}$$

Therefore, the upper bound of s can be derived as:

$$s \leq \frac{tf_1 - tf_2}{(k\frac{|D_2|}{avdl} - 1)tf_2 - (\frac{|D_2|}{avdl} - 1)tf_1}$$

where $tf_1 = 1 + ln(1 + ln(k \times c(t, D_2))), tf_2 = 1 + ln(1 + ln(c(t, D_2)))$. In order to get a sense of what the bound is exactly, consider a common case when $|D_2| = avdl$. We have

$$s \leq \frac{1}{k-1} \times (\frac{tf_1}{tf_2} - 1).$$

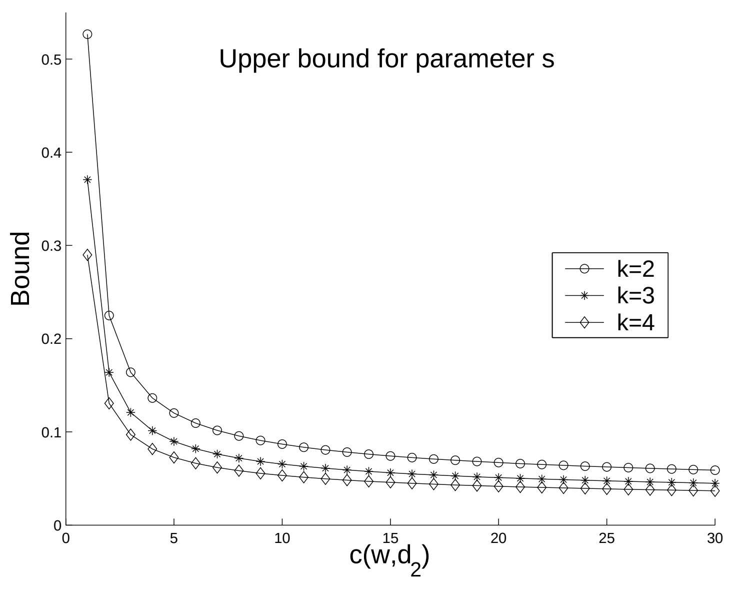


Figure 4.1: Upper bound of parameter s.

As shown in the Figure 4.1, the bound becomes tighter when k increases or when the term frequency is larger. This bound shows that in order to avoid over-penalizing a long document, a reasonable value for s should be generally small – it should be below 0.4 even in the case of a small k, and we know that for a larger k the bound would be even tighter. This analysis thus suggests that the performance can be bad for a large s, which is confirmed by our experiments.

4.1.2 Okapi

Okapi [43] is a highly effectiveness retrieval function that represents the classical probabilistic model. The function as presented in [56] is

$$S(Q,D) = \sum_{t \in Q \cap D} \left(ln \frac{N - df(t) + 0.5}{df(t) + 0.5} \times \frac{(k_1 + 1) \times c(t,D)}{k_1((1-b) + b\frac{|D|}{avdl}) + c(t,D)} \times \frac{(k_3 + 1) \times c(t,Q)}{k_3 + c(t,Q)} \right)$$

where k_1 (between 1.0-2.0), b (usually 0.75), and k_3 (between 0-1000) are constants.

The major difference between Okapi and other retrieval formulas is the possibly negative value of the IDF part, which has been discussed in [44]. It is trivial to show that if $df(t) >$

$N/2$, the IDF value would be negative.

When the IDF part is positive (which is mostly true for keyword queries), it is easy to see that Okapi method satisfies TFCs and LNCs. By considering a common case when $|D_2| = avdl$, the TF-LNC constraint is shown to be equivalent to $b \leq \frac{avdl}{c(t,D_2)}$. Since b is always smaller than 1, TF-LNC can be satisfied unconditionally. Moreover, we can show that TDC is unconditionally satisfied.

Although Okapi satisfies some constraints conditionally, unlike in the pivoted normalization method, the conditions do not provide any bound for the parameter b. Therefore, the performance of Okapi can be expected to be less sensitive to the length normalization parameter than the pivoted normalization method, which is confirmed by our experiments.

When the IDF part is negative, the Okapi formula would satisfy TDC but violate the TFCs, LNCs and TF-LNC, since matching an additional occurrence of a query term could mean decreasing the score. Since a negative IDF only happens when a query term has a very high document frequency (e.g., when the query is verbose), our analysis suggests that the performance of Okapi may be relatively worse for verbose queries than for keyword queries.

A simple way to solve the problem of negative IDF is to replace the original IDF in Okapi with the regular IDF in the pivoted normalization formula:

$$\sum_{t \in Q \cap D} \left(\frac{N+1}{df(t)} \times \frac{(k_1+1) \times c(t,D)}{k_1((1-b)+b\frac{|D|}{avdl}) + c(t,D)} \times \frac{(k_3+1) \times c(t,Q)}{k_3 + c(t,Q)} \right)$$

This modified Okapi satisfies all the defined constraints. We thus hypothesize that the modified Okapi would perform better than the original Okapi for verbose queries. As will be shown later, this is indeed true according to our experiment results.

Table 4.2: Constraint analysis results (Okapi)

Formula	TFCs	TDC	LNC1	LNC2	TF-LNC
Original	Cond.	Yes	Cond.	Cond.	Cond.
Modified	Yes	Yes	Yes	Yes	Yes

The results of analyzing the Okapi formula are summarized in Table 4.2. We distinguish two forms of the formula – the original formula and the one with a modified IDF part. The modification significantly affects the constraint analysis results as discussed above.

4.1.3 Dirichlet Prior

Dirichlet prior is one of the best performing language modeling approaches [77]. With a notation consistent with formulas above, the Dirichlet prior retrieval function is

$$S(Q,D) = \sum_{t \in Q \cap D} c(t,Q) \cdot ln(1 + \frac{c(t,D)}{\mu \cdot p(t|C)}) + |Q| \cdot ln\frac{\mu}{|D| + \mu}$$

$p(t|C)$ is similar to the document frequency $df(t)$, and it indicates how popular the term t is in the whole collection.

Table 4.3: Constraint analysis results (Dirichlet)

TFCs	TDC	LNC1	LNC2	TF-LNC
Yes	Yes	Yes	Cond	Yes

The results of analyzing the Dirichlet prior formula are summarized in Table 4.3. TFCs, TDC, LNC1 and TF-LNC are easily seen to be satisfied. So we only examine some of the non-trivial constraints.

The LNC2 constraint can be shown to be equivalent to $c(t, D_2) \geq |D_2| \cdot p(t|C)$, which is usually satisfied for content-carrying words. If all the query terms are discriminative words, long documents will not be over-penalized. Thus, compared to pivoted normalization, Dirichlet prior appears to have a more robust length normalization mechanism, even though none of them satisfies the LNC2 constraint unconditionally.

Table 4.4: Comparison between different retrieval formulas

Formula	**TFCs**	**TDC**	**LNC1**	**LNC2**	**TF-LNC**
Pivoted	Yes	Yes	Yes	C_1^*	C_2^*
Dirichlet	Yes	Yes	Yes	C_3	Yes
Okapi(original)	C_4	Yes	C_4	C_4	C_4
Okapi(modified)	Yes	Yes	Yes	Yes	Yes

4.1.4 Summary of Constraint Analysis Results

We have applied our seven constraints to three representative retrieval formulas. The results are summarized in Table 4.4, where a "Yes" means the corresponding model satisfies the particular constraint and a "C_x" means corresponding model satisfies the particular constraint under some particular conditions (irrelevant to parameter setting), and a "C_x^*" means the model satisfies the constraint only when the parameter is in some range. The specific conditions are

$$\begin{aligned}
C_1^* &\Leftrightarrow s \leq \frac{tf_1 - tf_2}{(k\frac{|D_2|}{avdl} - 1)tf_2 - (\frac{|D_2|}{avdl} - 1)tf_1} \\
C_2^* &\Leftrightarrow s \leq \frac{(h(c(t,D_1)) - h(c(t,D_2))) \times avdl}{(c(t,D_1) - c(t,D_2)) \times (1 + h(c(t,D_1)))} \\
C_3 &\Leftrightarrow c(t,D_2) \geq |D_2| \cdot p(t|C) \\
C_4 &\Leftrightarrow idf(t) \geq 0 \Leftrightarrow df(t) \leq N/2
\end{aligned}$$

Based on the results, we can make two interesting observations:

First, it is surprising that the original IDF part of Okapi formula causes the formula to violate almost all constraints, thus we may predict that the Okapi formula may have a worse performance for verbose queries.

Second, C_1 and C_2 provide an approximate bound for the parameters in the pivoted normalization method. In contrast, by checking the constraints, we have not found any particular bound for the parameter in Dirichlet Prior and Okapi. Therefore, we predict that the pivoted normalization method is more sensitive to the parameter setting than the other two methods.

4.2 Experiments

In the previous section, we have examined three representative retrieval formulas analytically. Based on the analysis, we propose some hypotheses about the performance for each retrieval formula. In this section, we test these hypotheses through carefully designed experiments. Our experiment results show that the proposed constraints can both explain the performance difference in various retrieval models and provide an approximate bound for the parameters in a retrieval formula.

4.2.1 Experiment Design

Retrieval performance can vary significantly from one test collection to another. We thus construct several very different and representative test collections using the existing TREC test collections.

To cover different types of queries, we follow [77] , and vary two factors: query length and verbosity, which gives us four different combinations : short-keyword (SK, keyword title), short-verbose (SV, one sentence description), long-keyword (LK, keyword list), and long-verbose (LV, multiple sentences). The number of queries is usually larger than 50. To cover different types of documents, we construct our document collections by varying several factors, including (1) the type of documents; (2) document length; (3) collection size; and (4) collection homogeneity. Our choice of document collection has been decided to be news articles (AP), technical reports (DOE), government documents (FR), a combination of AP, DOE, and FR (ADF), the Web data used in TREC8 (Web), the ad hoc data used in TREC7 (Trec7) and the ad hoc data used in TREC8 (Trec8). Table 4.5 shows some document set characteristics, including the number of queries used on the document set, the average

Table 4.5: Document set characteristic

	AP	**DOE**	**FR**	**ADF**	**Web**	**Trec7**	**Trec8**
#qry	142	35	42	144	50	50	50
#rel/q	103	57	33	126	46	93	95
size	491MB	184MB	469MB	1GB	2GB	2GB	2GB
#doc(k)	165K	226K	204K	437K	247K	528K	528K
#voc(k)	361K	163K	204K	700K	1968K	908K	908K
mean(dl)	454	117	1338	372	975	477	477
dev(dl)	239	58	5226	1739	2536	789	789
mean(rdl)	546	136	12466	1515	6596	1127	1325

number of relevant documents per query, the collection size, the number of documents, the vocabulary size, the mean document length, the standard deviation of document length, and the mean length of relevant documents.

The preprocessing of documents and queries is minimum, involving only Porter's stemming. We intentionally did not remove stop words for two reasons: (1) A truly robust model should be able to discount the stop words automatically; (2) Removing stop words would introduce at least one extra parameter (e.g. the number of stop words) into our experiments. On each test collection, for every retrieval method, we vary the retrieval parameter to cover a reasonably wide range of values. This allows us to see a complete picture of how sensitive each method is to its parameter.

4.2.2 Parameter Sensitivity

Based on the analysis in the previous section, we formulate the following hypotheses: (1) The pivoted normalization method is sensitive to the value of parameter s. The analysis of LNC2 suggests that the reasonable value for s should be generally smaller than 0.4 and the performance can be bad for a large s. (2) Okapi is more stable with the change of parameter b compared with the pivoted normalization method.

We now discuss the experiment results. First, let us consider the experiment result for the pivoted normalization method. As shown in Table 4.6, the optimal value of s to maximize

Table 4.6: Optimal s (for average precision) in the pivoted normalization method

	AP	**DOE**	**FR**	**ADF**	**Web**	**Trec7**	**Trec8**
lk	0.2	0.2	0.05	0.2	—	—	—
sk	0.01	0.2	0.01	0.05	0.01	0.05	0.05
lv	0.3	0.3	0.1	0.2	0.2	0.2	0.2
sv	0.2	0.3	0.1	0.2	0.1	0.1	0.2

average precision is indeed very small in all cases. Moreover, Figure 4.2 shows how the average precision is influenced by the parameter value in the pivoted normalization method on the AP document set and long-keyword queries; the curves are similar for all other data sets. Clearly when s is large, which causes the method not to satisfy the LNC2 constraint, the performance becomes significantly worse.

In contrast, we experiment with the Okapi method. Assume $k_1 = 1.2, k_3 = 1000$ [] and b changes from 0.1 to 1.0. Okapi is indeed more stable than the pivoted normalization (shown in Figure 4.2). By checking the constraints, we have not found any particular bound for the parameters in Okapi, which may explain why Okapi is much less sensitive to the parameter setting than the pivoted normalization method where the LNC2 constraint implies a concrete bound on parameter s.

In summary, the constraints generally can provide an empirical bound for the parameters in retrieval formulas and the performance would tend to be poor when the parameter is out of the bound.

4.2.3 Performance Comparison

We compare the performance of the three retrieval formulas through systematic experiments. Our goal is to see whether the experiment results are consistent with the analytical results based on formalized heuristics. We form the following hypotheses based on the constraint analysis:(1) For any query type, the Dirichlet prior method performs comparably to pivoted normalization method when the retrieval parameters are set to optimal values. (2) For

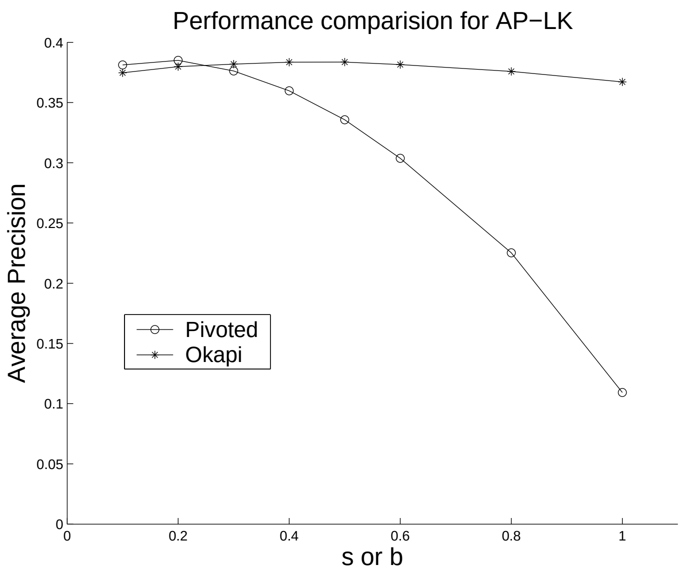


Figure 4.2: Performance Comparison between Okapi and Pivoted for AP-LK.

keyword queries, Okapi performs comparably to the other two retrieval formulas. (3) For verbose queries, Okapi may perform worse than others, due to its possible negative IDF part. As mentioned in the previous subsection, when IDF is negative, Okapi violates almost all the constraints. However, if we modify the Okapi formula by replacing the original IDF part with IDF part of the pivoted normalization method, the formula would satisfy almost all the constraints for any query type, therefore we hypothesize that the modified Okapi formula performs better than the original one for verbose queries.

In order to test these hypotheses, we run experiments over seven collections and four query sets by using the pivoted normalization method, the Dirichlet prior method, Okapi and the modified Okapi formula. We use average precision as the evaluation measure, and summarize the optimal performance for each formula in Table 4.7. The results show that for verbose queries, the performance of the Mod-Okapi is significantly better than that of Okapi; the p-values of the Wilcoxon signed rank test are all below 0.013.

We see that, indeed, for keyword queries, the performances of three retrieval formulas are comparable. However, for verbose queries, Okapi is much worse than others in most cases. We hypothesis that this is caused by the negative IDF scores on common words. This

Table 4.7: Comparison of optimal performance for four formulas.

		AP	**DOE**	**FR**	**ADF**	**Web**	**Trec7**	**Trec8**
lk	Pivoted	0.39	0.28	0.33	0.27	—	—	—
	Dirichlet	0.38	0.28	0.32	0.25	—	—	—
	Okapi	0.38	0.27	0.28	0.33	—	—	—
	Mod-Okapi	0.39	0.28	0.28	0.33	—	—	—
sk	Pivoted	0.23	0.18	0.19	0.22	0.29	0.18	0.24
	Dirichlet	0.22	0.18	0.18	0.21	0.30	0.19	0.26
	Okapi	0.23	0.19	0.23	0.19	0.31	0.19	0.25
	Mod-Okapi	0.23	0.19	0.23	0.19	0.31	0.19	0.25
lv	Pivoted	0.29	0.21	0.23	0.21	0.22	0.20	0.23
	Dirichlet	0.29	0.23	0.24	0.24	0.28	0.22	0.26
	Okapi	0.03	0.07	0.09	0.06	0.23	0.08	0.11
	Mod-Okapi	0.30	0.24	0.25	0.23	0.28	0.26	0.25
sv	Pivoted	0.19	0.10	0.14	0.14	0.21	0.15	0.20
	Dirichlet	0.20	0.13	0.16	0.16	0.27	0.18	0.23
	Okapi	0.08	0.08	0.08	0.09	0.21	0.09	0.10
	Mod-Okapi	0.19	0.12	0.16	0.14	0.25	0.16	0.22

hypothesis is verified by the modified Okapi. After replacing the IDF part in Okapi with the IDF part of the pivoted normalization formula, the performance is improved significantly for the verbose queries. See Figure 4.2 and Figure 4.3 for plots of these comparisons.

Figure 4.3 concludes that satisfying more constraints appears to be correlated with a better performance. Therefore, the proposed constraints provide a plausible explanation for the performance difference in various retrieval models, and suggest how we may improve a retrieval formula further.

4.3 Conclusions and Future Work

In this chapter, we study the problem of formalizing the necessary heuristics for good retrieval performance. Motivated by some observations on common characteristics of typical retrieval formulas, we formally define six basic constraints that any reasonable retrieval function should satisfy. These constraints correspond to some desirable intuitive heuristics, such

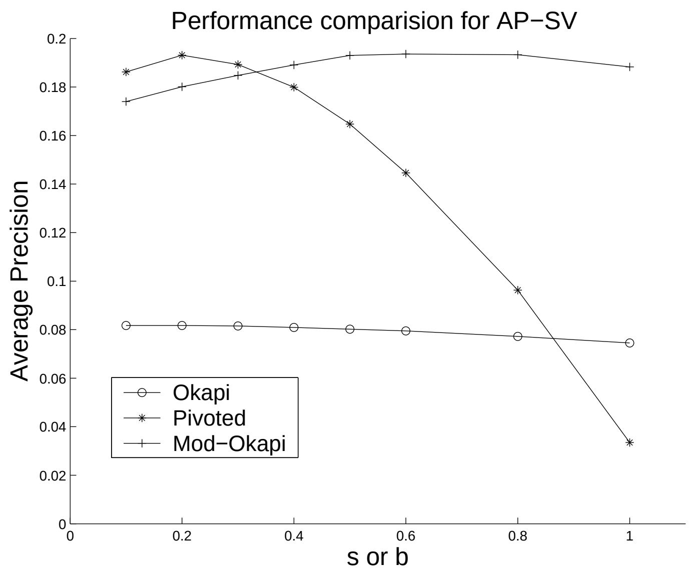


Figure 4.3: Performance Comparison between modified Okapi, Okapi and Pivoted for AP-SV.

as term frequency weighting, term discrimination weighting and document length normalization. We check these six constraints on three representative retrieval formulas analytically and derive specific conditions when a constraint is conditionally satisfied. The constraint analysis suggests many interesting hypotheses about the expected performance behavior of all these retrieval functions. We design experiments to test these hypotheses using different types of queries and different document collections. We find that in many cases the empirical results are indeed consistent with these hypotheses. Specifically, when a constraint is not satisfied, it often indicates non-optimality of the method. This is most evident from the analysis of Okapi formula, based on which we successfully predict the non-optimality for verbose queries. In some other cases, when a method only satisfies a constraint for a certain range of parameter values, its performance tends to be poor when the parameter is out of this range, which is evident in the analysis of the pivoted normalization and the Dirichlet prior. In general, we find that the empirical performance of a retrieval formula is tightly related to how well they satisfy these constraints. Thus the proposed constraints can provide a good explanation of many empirical observations (e.g., the relatively stable performance of the Okapi formula) and make it possible to evaluate any existing or new retrieval formula *analytically*, which is extremely valuable for testing new retrieval models. Moreover, when a constraint is not satisfied by a retrieval function, it also suggests a possible way to improve the retrieval formula.

There are many interesting future research directions based on this work.

First, since our constraints do not cover all the desirable properties, it would be interesting to explore additional necessary heuristics for a reasonable retrieval formula. This will help us further understand the performance behavior of different retrieval methods. The retrieval constraints that have been studied in this chapter are the most intuitive ones. Existing effective retrieval functions all satisfy the current constraints conditionally or unconditionally. It would be really interesting to find some constraints that are desirable but not satisfied by any existing retrieval functions. The current retrieval constraints mainly comes from our empirical observations. We also plan to find more retrieval constraints by using exploratory data analysis [18]. Our goal is to let data speaks for itself. We wish that the past data will "tell" us which retrieval factors affect the retrieval performance and how they affect. When we add more constraints to the framework, we would inevitably introduce some bias. We think it is OK to have such biased constraints as long as they are reasonable in most of the cases. We can test the reasonableness of a constraint by using some synthetic data sets. More constraints also make it possible that some constraints conflict with others. This is not necessarily a bad thing. The performance of retrieval functions seems to have reached the plateau, which might be explained by the conflict between constraints. So it would be interesting to study conflict theoretical properties of retrieval functions along a similar line to a related work on clustering algorithms [24].

Second, we will apply these constraints to many other retrieval models proposed in the literature [3] and different smoothing methods for language models as well [77]. Previous work [49, 79] has attempted to identify an effective retrieval formula through extensive empirical experiments, but the results are generally inconclusive with some formulas performing better under some conditions. Analysis of formalized retrieval constraints as explored in this thesis may shed some light on what these conditions are exactly.

Third, the proposed constraints are all binary, which makes it impossible to tell which implementation is better if all implementations satisfy the constraints. Thus, it would be interesting to study how to quantitatively evaluate how well a retrieval function implements the constraints, which is the focus of next chapter.

Finally, the fact that none of the existing formulas that we have analyzed can satisfy all the constraints unconditionally suggests that it would be very interesting to see how to develop new retrieval functions that satisfy all constraints, which presumably would perform better empirically than these existing methods. In Chapter 6, we discuss how to derive new retrieval functions using axiomatic approach.

Chapter 5

Empirical Diagnostic Evaluation

Although constraint analysis can pinpoint some differences between different retrieval functions, it would not help if the analyzed retrieval functions all satisfy the retrieval constraints. To address this limitation, we propose a novel general methodology to *empirically* diagnose the weaknesses of retrieval functions. Our main idea is to carefully design a set of diagnostic tests to amplify the differences among constraint implementations in retrieval functions. Specifically, we first define a set of relevance-preserving collection perturbation operators as the basic tools for diagnostic tests. Such collection perturbations make it possible to reveal the differences among retrieval functions more explicitly. We then present a common procedure for designing diagnostic tests for retrieval models based on the defined perturbation operators. Following the proposed procedure, we design a group of diagnostic tests to test different aspects of retrieval functions, including length variation robustness, term noise resistance, and appropriate balance of term frequency and length normalization. Experiment results demonstrate several benefits of the proposed diagnostic evaluation methodology. First, it can reveal several clear patterns that are originally hidden in the existing evaluation methodology. Second, it allows us to identify the strengths and weaknesses of retrieval functions. Finally, it provides guidance on how to modify a retrieval function to achieve better performance. After we make the modification based on the diagnostic test results, the modified retrieval functions outperform the corresponding original retrieval functions in most cases.

5.1 The Basic Idea

The defined constraints are all binary, in the sense that we know only whether a function satisfies a constraint or not. Thus constraint analysis is not sufficient if the analyzed retrieval function satisfies all the constraints, constraint analysis is not sufficient and we still need to rely on the existing evaluation methodology. A retrieval function is typically evaluated using standard test collections and evaluation measures such as the Mean Average Precision (MAP) and precision at 10 documents, which generally reflect the utility of a retrieval function. Unfortunately, such an evaluation methodology provides little *explanation* for the performance differences among retrieval functions. For example, comparing two retrieval functions based on MAP, we can only know which function gives an overall better ranking of documents on a particular data set, but it is hard to identify the underlying causes of such performance difference.

To address this limitation, we discuss a general methodology to *diagnose* the degree of constraint satisfaction of any retrieval function. Since satisfaction of constraints is related to the empirical performance of a retrieval function, such methodology allows us to diagnose the weaknesses and strengths of a retrieval function based on their empirical performance patterns. Our idea is best understood through the following medical domain analogy.

In medical domain, diagnosis is the process of recognizing a disease based on its symptoms. Medical instruments, such as medical thermometers, are necessary for diagnosis. Typically, based on the symptoms and medical records of a patient, a physician will perform various medical tests, such as measuring body temperature, with available medical instruments. Based on the results of the medical test, the physician would analyze the underlying causes of the disease and provide appropriate treatments for the patient.

By making this analogy (shown in Table 5.1), we can gain some insights about how to do retrieval function diagnosis: First, we need necessary "instruments" to perform diagnoses. We propose a set of operators for perturbing existing evaluation collections while preserving

Table 5.1: Medical diagnosis analogy

Medical Domain	**IR Domain**
patients	retrieval functions
illness	non-optimal performance
diseases	problems of heuristic implementation, causes of non-optimal performance
medical records	existing findings in IR
symptoms	empirical results
medical instruments	relevance-preserved collection perturbations
medical tests	tests for retrieval models
treatments for disease	better implementations of heuristics, modification for performance improvement

the relevance status of all documents. Such perturbations make it possible to enlarge the differences among retrieval functions and make it easier to observe the "symptoms" of existing retrieval functions (i.e., the problems of current heuristic implementations). Second, we present a common procedure for designing diagnostic tests for retrieval functions and propose some measures to help interpret the results of these tests. We then design several diagnostic tests targeting at testing specific aspects of a retrieval function. Finally, we perform the tests, analyze the results, identify the problems of heuristic implementations and provide explanations for the empirical performance of retrieval functions and "treatments" to further improve the retrieval function.

5.2 Collection Perturbations

One reason why existing evaluation methodology is not informative enough is because a test collection usually has a mixed set of documents with various characteristics. One possible solution is thus to perturb an existing evaluation collection to control or vary some characteristics. This would generate a series of perturbed collections, which can then be used to test a retrieval function. We hope the perturbed collections would allow us to see more meaningful differences between retrieval functions. With our medical domain analogy, these perturbations serve as our instruments to perform diagnostic tests for retrieval models.

Table 5.2: Basic Perturbation Operators

Name	**Semantic**	**Operator**
term addition	Add K occurrences of term t to document D	$aT(t, D, K)$
term deletion	Delete K occurrences of term t from document D	$dT(t, D, K)$
document addition	Add document D to the collection K times	$aD(D, K)$
document deletion	Delete document D form the collection	$dD(D)$
document concatenation	Concatenate document D_1 with D_2 K times,	$cD(D_1, D_2, K)$

We first introduce some new notations. $\mathcal{D}$ is the document set in the test collection, $\mathcal{D}_r$ is the relevant document set, and $\mathcal{D}_n$ is the non-relevant document set. $\mathcal{V}_n$ is the noise term vocabulary that does not include any terms contributing to the relevance for any queries. It means that $\forall t_n \in \mathcal{V}_n$, t_n can not affect the relevance status of document D.

A standard evaluation collection includes a document collection, a set of queries, and a set of relevance judgements indicating which documents are relevant to which queries. To leverage the relevance judgments in the existing test collections, we keep the topics and perturb only the documents, which means to perturb term statistics in documents (e.g., term occurrences), document statistics (e.g., document length), and collection statistics (e.g., number of documents). We define five basic operators for document perturbation, including term addition, term deletion, document addition, document deletion, and document concatenation; they are summarized in Table 5.2. Every operator has a parameter K to control the degree of perturbation. K can either be the same for all documents or vary according to term/document statistics, such as the occurrences of a term. All the basic operators can be combined to perform more sophisticated perturbations.

Since we do not want to recreate the judgements, we should keep the relevance status of documents during the perturbations. Following the definition of relevance used in TREC, we assume that any relevance evidence in a document makes the document relevant. We further assume that both queries and documents use "bag of term" representation. Not all the proposed basic operators can guarantee to maintain the relevance status of a document. For example, delete query terms from a relevant document could change the document to non-

Table 5.3: Relevance-preserving perturbations

Name	**Semantic**	**Perturbation**
relevance addition	Add a query term to a a relevant document	$aT(t, D, K)$, where $D \in \mathcal{D}_r, t \in Q$
noise addition	Add a noisy term to a document	$aT(t, D, K)$, where $D \in \mathcal{D}, t \in \mathcal{V}_n$
internal term growth	Add a term to a document that originally contains the term	$aT(t, D, K)$, where $D \in \mathcal{D}, t \in D$
document scaling	Concatenate D with itself K times	$cD(D, D, K)$, where $D \in \mathcal{D}$
relevant doc. concatenation	Concatenate two relevant documents K times	$cD(D_1, D_2, K)$, where $D_1, D_2 \in \mathcal{D}_r$
non-relevant doc. concatenation	Concatenate two non-relevant documents K times	$cD(D_1, D_2, K)$, where $D_1, D_2 \in \mathcal{D}_n$
noise deletion	Delete a term from a non-relevant document	$dT(t, D, K)$, where $D \in \mathcal{D}_n, t \in D$
document addition	Add a document to the collection	$aD(D, K)$
document deletion	Delete a document from the collection	$dT(D)$, where $D \in \mathcal{D}$.

relevant. Thus, what we need is *relevance-preserving* perturbations. A relevance-preserving perturbation is a perturbation where we have high confidence about the relevance status of each document after the perturbation. We define several *relevance-preserving* perturbations based on the proposed basic operators (summarized in Table 5.3). All these perturbations are intuitive. For example, a relevant document remains relevant if we add more query terms. Also, under the assumption that a document is relevant as long as part of it is relevant, adding noisy terms to any document would not change its relevance status. Furthermore, the relevance status of a document remains the same if we concatenate it with itself several times. Similarly, concatenating two relevant documents or two non-relevant documents would not affect the relevance status either. Note that in document concatenation, the changes of term occurrences are proportional to the document length.

5.3 Diagnostic Tests for IR Models

5.3.1 A Common Procedure

We now discuss how to design diagnostic tests for retrieval models.

First, we need to identify the aspects of retrieval functions that we want to test. For example, in the medical domain, a physician measures the body temperature of a patient to know whether the patient is having a fever. Similarly, for retrieval model diagnosis, we should also identify some desirable properties of a reasonable retrieval function for diagnosis, which is in line with the formalized retrieval constraints proposed in Chapter 3.

Second, we need to connect the properties that are to be diagnosed with the relevance-preserving perturbations and select appropriate perturbation operators. Once the perturbations are chosen for a particular property, we could use the perturbation parameter of the operators to control the degree of perturbation. For a possible perturbation parameter value, we can record the empirical performance of retrieval functions on the corresponding perturbed collections, and stop doing this when we get enough information (i.e., when we can observe clear performance differences among retrieval functions). This procedure allow us to obtain a performance curve like the one shown in Figure 5.1; it gives us a picture of how the performance changes as we impose more dramatic perturbation. In such a figure, the x-axis is always the value of perturbation parameter, and y-axis is a standard retrieval performance measure, which is always MAP in our experiments.

Note that the perturbation parameter K could be set in many different ways. Here we only consider two possibilities: (1) *constant growth*, where K is the same for all terms and documents; (2)*linear growth*, where K is proportional to some term statistics, such as $c(t, D)$, or document statistics, such as $|D|$. It is often hard to decide a reasonable range of perturbations. In this work, we increase K and stop when we observe the performance differences among retrieval functions or when K reaches a sufficiently large number empirically set. We plan to study more principled ways to set the parameter range in the future. There are

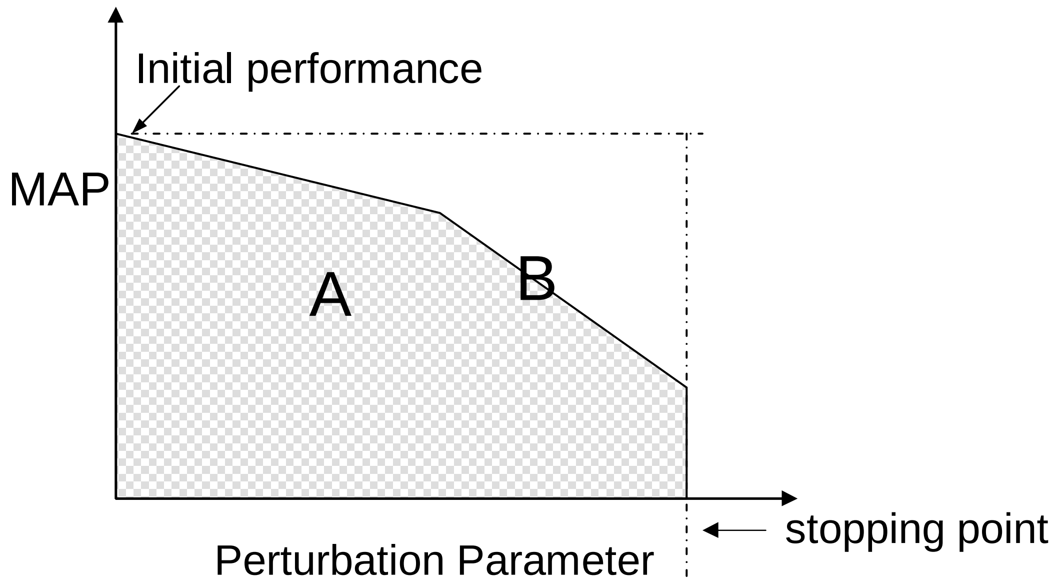


Figure 5.1: Example curve for the results of a diagnostic test

also different ways to select documents for perturbation: (1)*general collection perturbation*, where all the documents in the collection would be perturbed; (2)*sub-collection perturbation*, where only a subset of collection is perturbed. Note that we need to make sure that this choice should be consistent with the requirements of the relevance-preserving perturbations.

The perturbation results can be very useful to understand the behavior of a retrieval function. For example, a flat curve would mean that the function being tested is completely insensitive to such perturbation, while a dropping curve would mean that the function suffered from the perturbation. To interpret the results, we need to design measures to quantitatively evaluate perturbation results.

In general, measures can be defined based on the area under the curve or some extreme performance values (e.g., initial and end values or maximum and minimum values). Naturally, specific measures often depend on the property to be tested. In our study, we are most interested in how the performance degrades or increases as we increase the amount of perturbation. For this purpose, we define the following *performance ratio* (PR) measure:

$$PerformanceRatio \;=\; \frac{area\_under\_curve}{area\_under\_line\_through\_init\_point}$$

The PR value of the curve shown in Figure 5.1 can be computed by dividing area of shaded part A with the area of rectangle B. Intuitively, the PR value tells us the average amount of degradation or gain in performance after a perturbation. A high PR value indicates more gain in performance while a low PR value indicates more loss. The desirable PR value would depend on the specific perturbation, though in most of our experiments, a high PR value is better and suggests a more robust retrieval function. Note that the PR value can be larger than 1, which means that the retrieval performance increases as we increase the amount of perturbation.

Next we present three diagnostic tests designed by following this common procedure.

5.3.2 Length Variation Sensitivity Tests

Document length normalization is an important component in virtually all effective retrieval functions. To help understand a function's length normalization mechanism, we design tests to examine the sensitivity of a retrieval function to the length variance in the document collection. We use document scaling perturbation, *i.e.*, $cD(D, D, K)$, to perform the tests, because it changes the document length of each document, yet maintains the relative ratio of term occurrences. We design the following three different tests to examine the different aspects of the length variation. These three tests differ in how to set the value of perturbation parameter i.e., K.

Length variance reduction test (LV1): This test is to use the document scaling perturbation to make all the documents to have similar or identical length, and it would thus reduce the variance of document lengths. The test is defined as follows.

For every $D \in \mathcal{D}$, perform $cD(D, D, K)$

with $K = \frac{(1-\beta)\times|d|+\beta\times 1000000}{|d|}$ and $0 \le \beta \le 1$.

β is a parameter to control the degree of perturbation. When β is set to 0, all the documents have the original length. When β is set to 1, all the documents have the same length (i.e., number of terms in the documents is $1,000,000$).

Since more perturbation would deprive the retrieval function of any benefit of length normalization, the test result can indicate how effective the length normalization mechanism of a function is. A lower PR value indicates that the function could gain more through length normalization.

Length variance amplification test (LV2): This test is to amplify the differences in document lengths and make distribution of document lengths skewer. The test is defined as follows:

For every $D \in \mathcal{D}$, perform $cD(D, D, K)$,

where $K = |D| \times \beta$.

K is proportional to the original document length, which means that longer documents will grow much more rapidly compared with shorter ones. β is used to control the degree of perturbation. A larger β leads to skewer document length distribution. We expect a robust function to have a high PR value.

Length scaling test (LV3): This test is to concatenate all the documents with themselves K times, where K is same for all the documents. In this way, the length variance would change but the relative length ratio remains the same.

The test has the same intuition as the LNC2 constraint proposed in Chapter 3. Thus, if a retrieval function achieves a higher PR value, it means that the function does a better job to avoid over-penalizing long documents.

5.3.3 Term Noise Resistance Tests

A robust retrieval function should also be resistant to term noise, i.e., the terms that do not contribute to the relevance of a document ($\forall t_n \in \mathcal{V}_n$). We assume that a document is relevant if it contains some relevant evidence, so a reasonable retrieval function is expected to be resistant to the addition of term noise. We design the following test to examine the term noise resistance of a retrieval function.

Noise addition test (TN): This test is to add noise (i.e., non-relevant terms) to documents. We use the*noise addition perturbation* operator as follows:

For every $D \in \mathcal{D}$, perform $aT(t_n, D, K)$

where $t_n \in \mathcal{V}_n$ and K is a parameter.

There are two variations: (1)constant growth: K is a constant, i.e., we add the same number of noisy terms to all documents; and (2)Linear growth: $K = \beta \times |D| - \sum_{t \in Q} c(t, D)$, $\beta > 1$, i.e., the number of added noisy terms is linear to the original document length. The test has the same intuition as the LNC1 constraint. Thus, a high PR value indicates that the function is reasonable in penalizing long documents.

An alternative way to examine noise resistance would be to design a test to remove noise from documents. Unfortunately, this is not easy since non-query terms are also possible to contribute to the relevance of a document, we can not remove any terms from relevant documents. We leave this as part of our future work.

5.3.4 TF-LN Balance Tests

TF (term frequency) and LN (length normalization) are two important heuristics that are often coupled together in retrieval models due to the need for TF normalization. We design three tests to examine the ability of a retrieval function to balance TF and LN. The main idea is to increase the occurrences of the query terms that are already in the documents. The following three tests differ in how to pick the query terms whose occurrences are to be increased.

Single query term growth test (TG1): We increase the term occurrence for only one random query term, and use the internal term growth perturbation as follows:

```
t is a random query term, for every $D \in \mathcal{D}$
    if $t \in D$, perform $aT(t, D, K)$.
```

This test is designed to increase term occurrence of one query term so that a query term will *dominate* in a document. A retrieval function with a higher PR value for this test is more robust against such dominance and favors documents containing more distinct query terms.

Majority query term growth test (TG2): We can increase the term occurrences for all but one random query term. Test could be defined as follows:

```
t is a random query term, for every $D \in \mathcal{D}$
    for every $t' \in Q - \{t\}$, if $t' \in D$, perform $aT(t', D, K)$.
```

The test is designed to increase term occurrences for majority query terms so that only one query term will be less dominant in a document. Obviously this test is only meaningful for queries with at least two terms, and in the case when there are exactly two terms, it is the same as the previous test. A higher PR value indicates that the function is more robust against such majority dominance and favors documents containing more of the query terms (i.e., larger sum of all query term occurrences).

All query term growth test (TG3): We perform the internal term growth perturbation for all query terms :

```
For every $D \in \mathcal{D}$
```

Table 5.4: Tests and corresponding interpretations

Test	**Interpretation of higher value**	**Tag**
length variance reduction	have less gain on length normalization	LV1
length variance amplification	be more robust to larger document variance	LV2
length scaling	better at avoiding over-penalizing long documents	LV3
term noise addition	penalize long documents more appropriately	TN
single query term growth	favor documents with more distinct query terms	TG1
majority query term growth	favor documents with more query terms	TG2
all query term growth	balance tf and LN more appropriately	TG3

for every $t \in Q$, if $t \in D$, perform $aT(t, D, K)$.

This test is to examine whether the increase of tf can compensate for the score loss caused by the length penalization. A retrieval function with higher PR value can balance the TF and LN parts better.

All the proposed tests and their interpretations are summarized in Table 5.4.

5.4 Experiments

5.4.1 Setup

Following the setup in previous section, we use the same three retrieval functions as used in Chapter 4 for diagnosis: pivoted normalization (Piv.) [57, 56], Okapi (Ok.) [43, 46], and Dirichlet prior (Dir.) [77]. To make the implementation of heuristics clearer, we rewrite these retrieval functions as follows.

$$\begin{aligned}
Piv.: S(Q,D) &= \sum_{t \in Q \cap D} c(t,Q) \times IDF_{Piv}(t) \times \frac{TF_{Piv}(t,D)}{LN_{Piv}(D)} \\
Ok.: S(Q,D) &= \sum_{t \in Q \cap D} QTF_{Ok}(t) \times IDF_{Ok}(t) \times TFLN_{Ok}(t,D) \\
Dir.: S(Q,D) &= \sum_{t \in Q \cap D} c(t,Q) \times TFIDF_{Dir}(t,D) - |Q| \times LN_{Dir}(D)
\end{aligned}$$

Table 5.5: Optimal performance of representative functions

	trec7	trec8	wt2g	ap88-89	doe	fr88-89
Pivoted	0.176	0.244	0.288	0.227	0.179	0.218
Okapi	0.186	0.251	0.310	0.226	0.185	0.230
Dirichlet	0.186	0.257	0.302	0.224	0.180	0.202

where

$$\begin{aligned}
IDF_{Piv}(t) &= ln(\frac{N+1}{df(t)}) \\
TF_{Piv}(t,D) &= 1 + ln(1 + ln(c(t,D))) \\
LN_{Piv}(D) &= 1 - s + s\frac{|D|}{avdl} \\
QTF_{Ok}(t) &= \frac{(k_3+1) \times c(t,Q)}{k_3 + c(t,Q)} \\
IDF_{Ok}(t) &= ln\frac{N - df(t) + 0.5}{df(t) + 0.5} \\
TFLN_{Ok}(t,D) &= \frac{(k_1+1) \times c(t,D)}{k_1((1-b) + b\frac{|D|}{avdl}) + c(t,D)} \\
TFIDF_{Dir}(t,D) &= ln(1 + \frac{c(t,D)}{\mu \times p(t|C)}) \\
LN_{Dir}(D) &= ln(1 + \frac{|D|}{\mu})
\end{aligned}$$

We perform diagnostic tests over six data sets used in previous section. These three functions perform very similarly when optimized on these data sets as shown in Table 5.5. We cannot understand much how these functions differ just based on these MAP values. We will show that diagnostic tests are able to help us understand their underlying differences, diagnose their weaknesses and improve their retrieval performance. In these tests, the parameters of these retrieval functions are set to the optimal values as used in Table 5.5.

Table 5.6: Results of length variation robustness tests

Test	Pivoted	Okapi	Dirichlet	Desirable Value and Interpretation
LV1	0.945	**0.927**	0.943	Low $\rightarrow$ better implementation of LN
LV2	0.873	0.889	**0.908**	High $\rightarrow$ more robust in a collection with higher length variance
LV3	0.939	**0.953**	0.892	High $\rightarrow$ better at avoiding over-penalizing long documents

5.4.2 Results of Diagnostic Tests

Length variation sensitivity tests

Table 5.6 shows the results of three length variation sensitivity tests. Every value is the average PR (i.e., performance ratio) on six data sets. For the variance reduction test (i.e., LV1), pivoted has the highest PR value, which means that it is least sensitive to this test. On the other hand, okapi has the lowest PR value, which means that it has lost the most when we "turned off" its length normalization part, indicating that the length normalization part of Okapi is implemented more reasonably than other functions.

For the length variance amplification test (i.e., LV2), Dirichlet has the highest PR value, which means that it is more robust if we increase the length variances in the collection. Thus, it means that Dirichlet prior might be the best choice if the document lengths varies a lot in the collection.

For the length scaling test (i.e., LV3), okapi has the highest PR value, indicating that it is the most robust retrieval function for this test. Since this test has the same intuition as the LNC2 constraint, it can be regarded as a test to examine how well a retrieval function avoids over-penalizing the long documents. Thus, the lower PR values of Dirichlet and pivoted indicate that these two functions might not do a good job at avoiding over-penalizing the long documents.

To further verify our results of LV3, we design and perform two additional length scaling tests. Instead of scaling all documents, we conduct two tests where we scale only non-relevant documents (i.e., LV3-rel) and only relevant documents (i.e., LV3-nonrel), respectively. The

Table 5.7: Results of additional length scaling tests

	Pivoted	Okapi	Dirichlet	Desirable value and interpretation
LV3-nonrel	0.257	0.318	**0.675**	Low → better at avoiding over-penalizing long documents
LV3-rel	0.774	**1.87**	1.47	High → balance TF and LN better

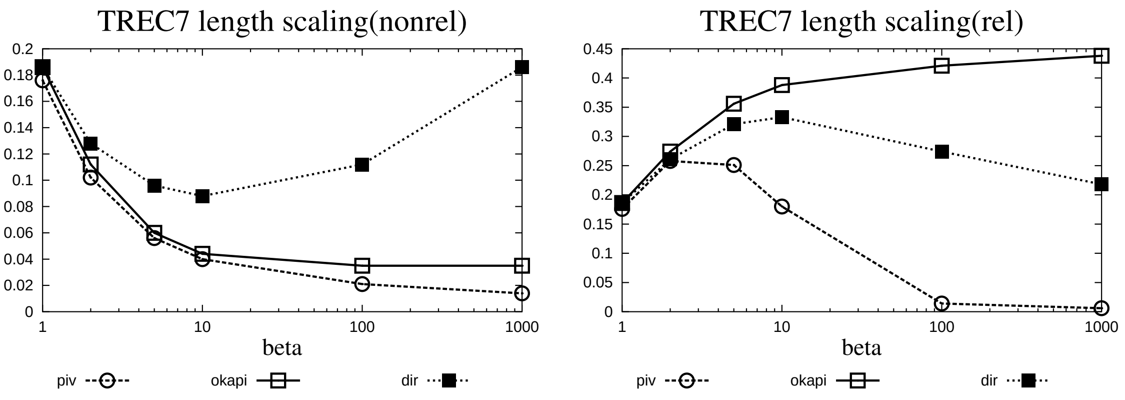


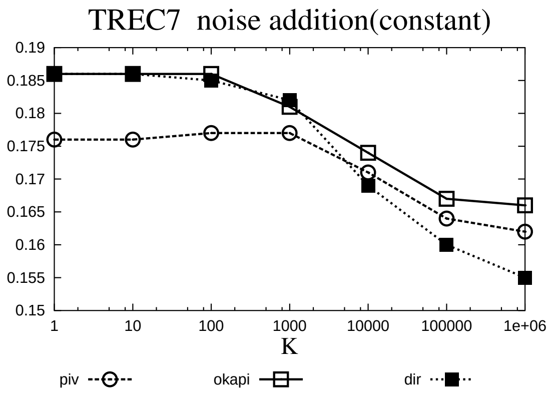
Figure 5.2: Additional length scaling tests:LV3-nonrel(Left), LV3-rel(Right)

results are shown in Table 5.7 and Figure 5.2. In LV3-nonrel test, all the non-relevant documents becomes longer. We would expect that a retrieval function that penalizes long documents more harshly to have a higher PR value. The highest PR value of Dirichlet indicates that it penalizes long documents more harshly than the other two functions. In LV3-rel test, both term frequency and document length grows in all relevant documents. We expect that a retrieval function that balances TF and LN well would get a higher PR value. The lowest PR value of pivoted indicates that it does not balance the growth of TF and LN as well as the other functions.

Term noise resistance tests

Table 5.8 and Figure 5.3 show the results of term noise resistance tests where the noisy terms are added to all the documents. The lowest PR value of Dirichlet indicates that Dirichlet does not penalize long documents as appropriately as others.

To further understand the results, we design one additional test (i.e., TN-nonrel). Instead



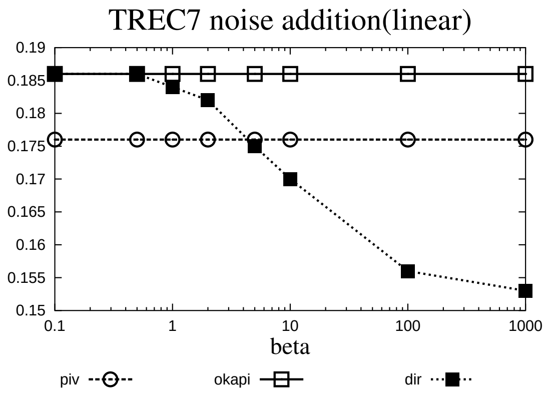
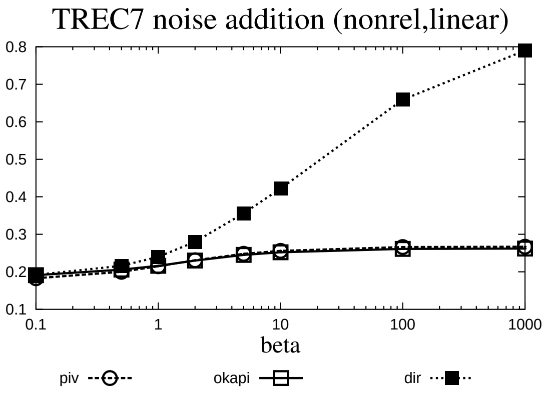


Figure 5.3: Term noise addition tests (TN)



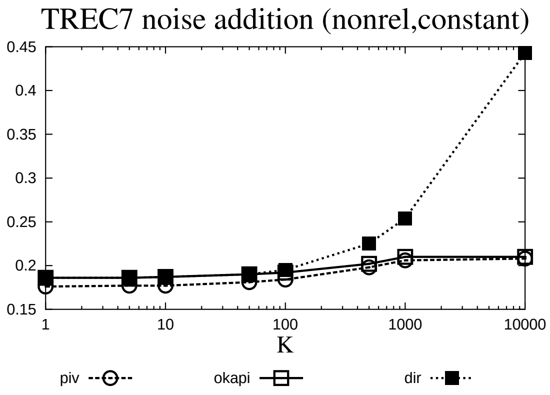


Figure 5.4: Additional term noise tests (TN-nonrel)

of performing the test on all the documents, we perform it only on non-relevant documents. Thus, when we do more perturbation, the length of a non-relevant document would become longer, and we expect that a retrieval function penalizing long documents more harshly would perform much better when we do more perturbation. Figure 5.4 shows the results for TREC7. The curve for other data sets are similar. The performance of Dirichlet grows more quickly, which means that it penalizes the long documents more harshly, which is consistent with our findings in non-relevant document length scaling test (i.e., LV3-nonrel).

Table 5.8: Results of noise resistant tests

	Pivoted	Okapi	Dirichlet	Desirable value and interpretation
TN (constant)	**0.973**	0.969	0.954	High → penalize long documents better
TN (linear)	**1**	**1**	0.916	High → penalize long documents better
TN-nonrel (constant)	1.11	1.12	**1.30**	Low → better at avoiding over-penalizing long documents
TN-nonrel (linear)	1.31	1.49	**2.27**	Low → better at avoiding over-penalizing long documents

TF-LN balance tests

The results for TF-LN balance tests are summarized in Table 5.9. For single query term growth test, pivoted and okapi have higher PR values than Dirichlet, indicating that they favor documents matching more distinct query terms. From the left plot in Figure 5.5, we see that Dirichlet performs differently from the other two.

For majority query growth test, Dirichlet has the highest PR, indicating that it favors the documents matching more query terms (i.e., larger sum of term occurrences for all the query terms in the document). The right plot in Figure 5.5 also shows that Dirichlet performs differently from the other two. It shows that Dirichlet starts decreasing when we have more perturbations (i.e. the documents are longer).

Based on the analysis from previous length-related tests, we know that Dirichlet tends to penalize long documents more harshly, which may be the cause for the decrease. To understand the results better, we perform another sets of tests, i.e., after performing query term growth tests, we perform length variance reduction test again to make all the document lengths equal. In this way, we can factor out the effect of length normalization. The results are shown in Figure 5.6, which is consistent with the previous observations. Therefore, we can make the conclusion that the behavior of TF part in Dirichlet is quite different from that part in Okapi and pivoted; both ways have their own advantages.

For all query term growth test, the results for constant growth and linear growth are not consistent as shown in Figure 5.7 and Table 5.9. In linear growth test, pivoted has

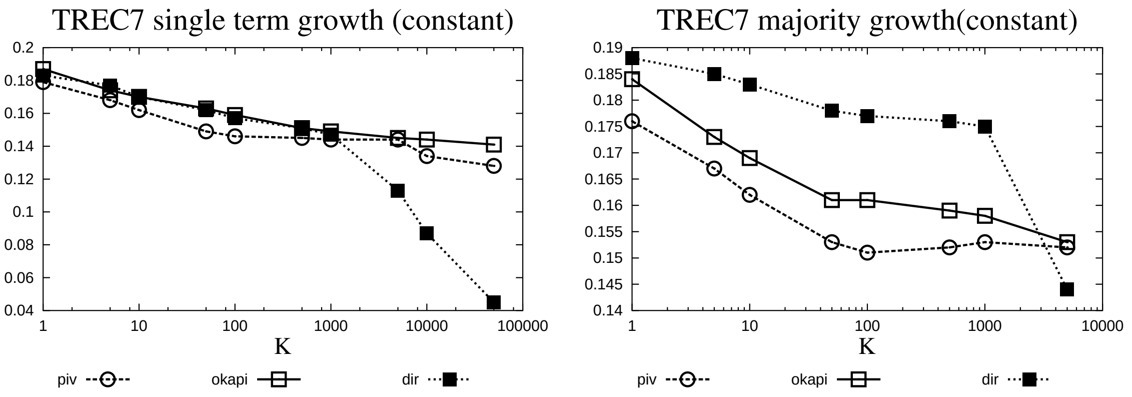


Figure 5.5: Single and majority term growth tests

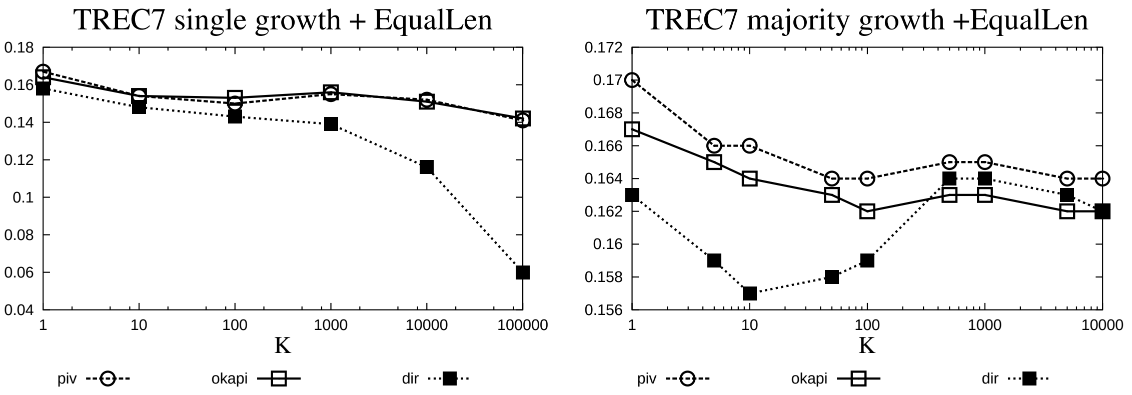
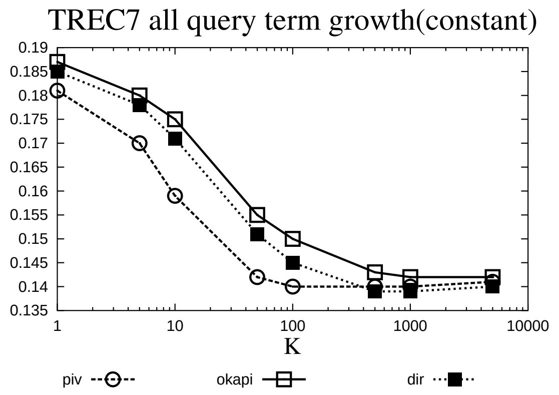


Figure 5.6: Single and majority term growth + Equal Len

the smallest PR value, which means that it can not balance TF and LN very well in these cases, because the increase of TF in pivoted can not compensate for the penalty caused by the document length. This observation is also consistent with our analysis of the relevant document length scaling test.

5.4.3 Analysis of Results

All the results presented in the previous subsection clearly demonstrate that the proposed diagnostic tests have the ability to pinpoint the weaknesses and strengths of the three functions. Here we briefly summarize our findings in Table 5.10. Instead of presenting results measured by PR, we give a confidence score to every pair-wise comparison. The confidence



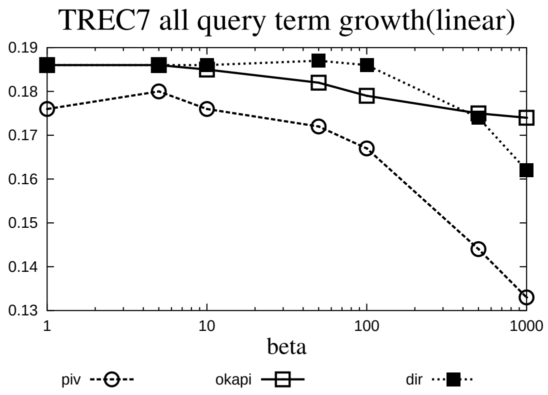


Figure 5.7: All query term growth tests

Table 5.9: Results of TF-LN balance tests (Larger value is better)

		Pivoted	Okapi	Dirichlet
single (TG1)	constant	0.860	**0.868**	0.785
	linear	0.874	**0.913**	0.865
majority (TG2)	constant	0.853	0.848	**0.869**
	linear	0.877	0.855	**0.944**
all (TG3)	constant	**0.799**	0.796	0.783
	linear	0.890	0.936	**0.962**

score is computed by the percentage of the data sets supporting the conclusion. For example, if 5 out of 6 data sets show that A performs better than B for test T, we have 83.3% confidence to say that A performs better than B for test T.

Comparing the findings from constraint analysis (as shown in Table **??**) with those from diagnostic tests (as shown in Table 5.10), we observe that many findings from these two strategies are consistent. First, constraint analysis shows that Okapi is the only retrieval function that satisfies both LN constraints, which is consistent with the results of LV1 test, i.e., the implementation of LN in Okapi is better. Second, Dirichlet is diagnosed to over-penalize long documents based on both constraint analysis (i.e., LNC2) and diagnostic tests (i.e., LV3, LV3-nonrel, TN and TN-nonrel). Finally, Pivoted does not balance the TF and LN well based on TF-LNC, LNC2, LV3-rel test, and TG3 test.

However, diagnostic tests could provide additional information that can not be found using constraint analysis. For example, we could identify the unique advantage of Dirichlet, i.e., it performs better when the document collection has larger length variance. Furthermore, constraint analysis can not identify the poor implementations of TF in the analyzed retrieval functions. On the contrary, diagnostic tests could identify the unique strengths and weaknesses of TF implementation in these retrieval functions.

Based on the results from constraint analysis and diagnostic tests, we summarize the weaknesses of every function in Table 5.11.

5.4.4 Improving Retrieval Functions

We now discuss how to modify existing retrieval functions based on the results of diagnostic tests, and compare the performance (i.e., MAP) of the modified functions with the existing retrieval functions. In all the results tables, ‡ and † indicate that the improvement is statistically significant according to Wilcoxon signed rank test at the level of 0.05 and 0.1 respectively.

Modifying LN Implementations

Both constraint analysis and diagnostic tests indicate that pivoted and Dirichlet suffer the problem of penalizing long documents harshly. To solve this problem, we modify them in the following ways.

$$\begin{aligned} MPln : S(Q,D) &= \sum_{t \in Q \cap D} c(t,Q) \times \frac{TF_{Piv}(t,D) \times IDF_{Piv}(t)}{LN_{Piv}(D)^{\lambda}} \\ MDln : S(Q,D) &= \sum_{t \in Q \cap D} c(t,Q) \times TFIDF_{Dir}(t,D) - |Q| \times LN_{Dir}(D)^{\lambda} \end{aligned}$$

where $0 < \lambda \leq 1$. The performance of the modified functions is reported in Table 5.12. It shows that such length normalization modification improves the performance.

Table 5.10: Summary of diagnostic results (? represents uncertainty, >> means more desirable)

Tests		**P. vs. D.**	**O. vs. D.**	**O. vs. P.**	Findings
LV1		$D >> P$ (66.7%)	$O >> D$ (83.3%)	$O >> P$ (100%)	Ok.: has better implementation of LN
LV2		$D >> P$ (83.3%)	$D >> O$ (83.3%)	$O?P$	Dir.: performs better in high-variance collections
LV3		$P >> D$ (83.3%)	$O >> D$ (100%)	$O >> P$ (66.7%)	Dir.: over-penalizes long documents
LV3-nonrel		$P >> D$ (100%)	$O >> D$ (100%)	$P >> O$ (100%)	Dir.: over-penalizes long documents
LV3-rel		$D >> P$ (100%)	$O >> D$ (83.3%)	$O >> P$ (100%)	Piv.: does not balance TF and LN well
TN	constant	$P >> D$ (83.3%)	$O >> D$ (66.7%)	$P > O$ (66.7%)	Dir.: over-penalizes long documents
	linear	$P >> D$ (100%)	$O >> D$ (100%)	$P = O$ (100%)	
TN -nonrel	constant	$P >> D$ (83.3%)	$O >> D$ (100%)	$P >> O$ (66.7%)	Dir.: over-penalizes long documents
	linear	$P >> D$ (100%)	$O >> D$ (100%)	$P >> O$ (66.7%)	
TG1	constant	$P >> D$ (100%)	$O >> D$ (100%)	$O >> P$ (66.7%)	Ok.: favors documents with more distinct query terms
	linear	$P >> D$ (66.7%)	$O >> D$ (83.3%)	$O >> P$ (66.7%)	
TG2	constant	$D >> P$ (66.7%)	$D >> O$ (83.3%)	$P >> O$ (66.7%)	Dir.: favors documents with more query terms
	linear	$D >> P$ (100%)	$D >> O$ (100%)	$P >> O$ (66.7%)	
TG3	constant	$D?P$	$O >> D$ (66.7%)	$O >> P$ (66.7%)	Piv.: does not balance TF and LN well
	linear	$D >> P$ (100%)	$D >> O$ (66.7%)	$O >> P$ (66.7%)	

Table 5.11: Weaknesses of functions and supporting constraints/tests

	Weaknesses	Constraints	Diagnostic tests
Piv.	can not balance TF and LN well	TF-LNC	LV3 (rel), TG3
	does not favor documents with more query terms	—	TG2
Ok.	does not favor documents with more query terms with more query terms	—	TG2
Dir.	penalizes long documents more harshly	LNC2	LV3, TN, LV3-nonrel TN-nonrel
	does not favor documents with more distinct query terms	—	TG1

Table 5.12: Improvement of LN modifications

	trec7	**trec8**	**wt2g**	**ap88-89**	**doe**	**fr88-89**
Piv.	0.176	0.244	0.288	0.227	0.179	0.218
MPln	0.181	0.250‡	0.306‡	0.227	0.180†	0.231
Dir.	0.186	0.257	0.302	0.224	0.180	0.202
MDln	0.187	0.259†	0.318‡	0.225	0.181‡	0.205

Modifying TF Implementations

It is unclear which TF implementation in the analyzed retrieval functions is more reasonable through constraint analysis. However, as we discussed early, the results of diagnostic tests allow us to identify the unique features in the TF implementations, and provide hints on how to make the current implementations more reasonable.

The diagnostic results show that the TF implementation of Dirichlet and that of Okapi (or Pivoted) represent two extreme cases: one is to favor documents with more query terms (i.e., larger sum of all query term occurrences), one is to favor documents with more distinct query terms. An ideal TF can be hypothesized to lie in somewhere between the two extremes. Based on this intuition, we heuristically modify the pivoted and Dirichlet as follows.

$$\begin{aligned} MPtf1 : S(Q,D) &= \sum_{t \in Q \cap D} c(t,Q) \times \frac{tfidf1(t,D)}{LN_{Piv}(D)} \\ MPtf2 : S(Q,D) &= \sum_{t \in Q \cap D} c(t,Q) \times \frac{tfidf2(t,D)}{LN_{Piv}(D)} \end{aligned}$$

Table 5.13: Improvement of TF modifications

	trec7	**trec8**	**wt2g**	**ap88-89**	**doe**	**fr88-89**
Piv.	0.176	0.244	0.288	0.227	0.179	0.218
MPtf1	0.182	0.248	0.293	0.227	0.183	0.218
MPtf2	0.182	0.250	0.296†	0.227	0.187‡	0.216
Dir.	0.186	0.257	0.302	0.224	0.180	0.202
MDtf1	0.187†	0.260†	0.310‡	0.228‡	0.183†	0.227‡
MDtf2	0.188†	0.255	0.310‡	0.229‡	0.182‡	0.213‡

$$\begin{aligned} MDtf1 : S(Q,D) &= \sum_{t \in Q \cap D} c(t,Q) \times tfidf1(t,D) - |Q| \times LN_{Dir}(D) \\ MDtf2 : S(Q,D) &= \sum_{t \in Q \cap D} c(t,Q) \times tfidf2(t,D) - |Q| \times LN_{Dir}(D) \end{aligned}$$

where

$$\begin{aligned} tfidf1(t,D) &= \alpha \times TF_{Piv}(t,D) \times IDF_{Piv}(t) + (1-\alpha) \times TFIDF_{Dir}(t,D) \\ tfidf2(t,D) &= \alpha \times TF_{Ok}(t,D) \times IDF_{Piv}(t) + (1-\alpha) \times TFIDF_{Dir}(t,D) \\ TF_{Ok}(t,d) &= \frac{2.2 \times c(t,D)}{1.2 + c(t,D)} \end{aligned}$$

and $\mu = 2000$ in $TFIDF_{Dir}(t,d)$, $0 \leq \alpha \leq 1$ and other notations are the same as explained at the beginning of the section.

The optimal performance of the modified retrieval functions is reported in Table 5.13. The results show that the modification can improve the performance, and tfidf2 usually performs better than tfidf1.

Additivity of modified TF and LN implementations

Since both LN and TF modifications are effective and tfidf2 is better, we can combine them in the following way.

Table 5.14: Additivity of TF and LN modifications

	trec7	**trec8**	**wt2g**	**ap88-89**	**doe**	**fr88-89**
Piv.	0.176	0.244	0.288	0.227	0.179	0.218
Dir.	0.186	0.257	0.302	0.224	0.180	0.202
Ok.	0.186	0.251	0.310	0.226	0.185	**0.230**
MPtf2ln	0.186	0.256	0.316	0.227	**0.187**	**0.230**
MDtf2ln	**0.190‡**	**0.262‡**	**0.321‡**	**0.229**	0.184	0.224

$$\begin{aligned} MPtf2ln : S(Q,D) &= \sum_{t \in Q \cap D} c(t,Q) \times \frac{tfidf2(t,D)}{LN_{Piv}(D)^{\lambda}} \\ MDtf2ln : S(Q,D) &= \sum_{t \in Q \cap D} c(t,Q) \times tfidf2(t,D) - |Q| \times LN_{Dir}(D)^{\lambda} \end{aligned}$$

The performance of combining ln and tfidf2 is reported in Table 5.14. The additivity of performance improvement of LN and TF comes from the fact that they capture different weaknesses in the retrieval function. Indeed, the modified retrieval functions outperform the optimized existing retrieval functions in most cases.

Although more parameters are introduced in the modified functions, the proposed diagnostic tests provide us guidance on where we should introduce additional parameters. Without such methodology, it is really hard to know how to modify a retrieval function to improve the retrieval performance in a principal way. We plan to study the sensitivity of these parameters and find alternative way to address the weaknesses without introducing extra parameters in the future.

The TF part and LN part in Okapi function are integrated together, which makes it harder to address the weaknesses of each component separately. We do not explore how to address the weaknesses of Okapi in the thesis, and leave it as future work. However, the new retrieval functions, that are derived using the proposed diagnostic evaluation methodology, outperform Okapi in most cases as shown in Table 5.14.

5.5 Conclusions and Future Work

In this chapter, we propose a novel general evaluation methodology to *empirically* diagnose weaknesses and strengths of retrieval models. We formally define a set of relevance-preserved collection perturbation operators, which can enlarge the performance difference and make it easier to pinpoint the weaknesses and strengths of retrieval functions. These operators serve as basic tools for us to perform diagnostic tests. We present a common procedure to design the diagnostic tests for retrieval models, where we connect the retrieval constraints with perturbation operators. Following the procedure, we design three sets of diagnostics tests, and perform the tests on six data sets. Experiments show that the proposed methodology can (1) identify the weaknesses and strengths of a retrieval function, (2) evaluate how well a retrieval function implements retrieval constraints, (3) explain the empirical differences among retrieval functions, and (4) give hints on how a retrieval function should be modified to further improve the performance. The modified retrieval functions based on diagnostic test results have been shown to outperform three representative retrieval functions in most cases. We further show that the empirical evaluation results are consistent with the analytical evaluation results as shown in Chapter 4, but are able to find more weaknesses which can not be identified through constraint analysis.

There are many interesting future research directions based on this work. First, it will be interesting to design more diagnostic tests to discover more interesting characteristics of different retrieval functions. The proposed diagnostic tests focus on only several basic retrieval heuristics. But if we could test more aspects of a retrieval function and found the weaknesses in these aspects, the function presumably would perform better empirically. Second, we could explore other measures to evaluate the results of diagnostic tests. The proposed PR measure is only one possible choice, and may not be the best one to quantitatively measure how well a retrieval function implements the corresponding retrieval constraints. Thus, it is interesting to study more measures in order to see which one is more reasonable.

Chapter 6

Derivation of New Retrieval Functions

The axiomatic approach offers a new way of developing retrieval models. In the previous chapters, we have shown how the constraint analysis and perturbation tests can provide insights about how to modify a retrieval function to improve performance. In this chapter, we study more systematically how to exploit the axiomatic approach to develop new retrieval functions.

The basic idea is to search in a space of candidate retrieval functions for one that can satisfy a set of reasonable retrieval constraints. The assumption is that if a retrieval function satisfies all our constraints, it would likely be effective empirically. In the previous chapters, we propose to define retrieval constraints by formalizing the retrieval heuristics. The remaining challenge in developing operational retrieval models using such an axiomatic approach is how to appropriately define the search space for retrieval formulas. To constrain the search space, we assume a "bag-of-terms" representation of queries and documents and propose to define a retrieval function *inductively*. Based on such a definition, a retrieval function can be decomposed into three components, referred to as *Primitive weighting function*, *Query growth function* and *Document growth function*, respectively. Thus searching for a good retrieval function boils down to searching for a good formula for each of these three functions in our constrained search space. The inductive definition scheme provides a common basis to analytically compare different retrieval functions. We compare and analyze three representative existing retrieval functions in this way and find that they share some commonalities in their primitive weighting functions and query growth functions, but they generally differ in the document growth function. The analysis provides an interpretation of the three component functions of the inductive definition scheme. We further generalize these specific component functions to derive new retrieval formulas within the axiomatic framework. We use the intuitive retrieval constraints proposed in the previous section and the technique of exploratory data analysis [18, 20] to constrain the choices for the three component functions and derive several new retrieval functions. We implement and test these new functions with a number of representative test sets. The experiment results show that the derived new functions are more robust and less sensitive to parameter settings than the existing retrieval functions with comparable optimal performance.

6.1 Inductive Function Space

One way to help us search through the function space efficiently is to define a retrieval function inductively. We start from the base case, where both the document and query only contain one term. In this case, the relevance score of the document for the given query is narrowed down to whether the term in the document matches the query term. When involving longer documents or longer queries, we further compute relevance score inductively by gradually adding all the terms in. We now present a formal definition of retrieval functions in this way, with the same notation introduced in the previous chapters.

Base Case: Assume $Q = \{q\}$ and $D = \{d\}$.

$$S(Q, D) = f(q, d) = \begin{cases} weight(q) = weight(d) & q = d \\ penalty(q, d) & q \neq d \end{cases} \tag{6.1}$$

Function f, referred to as the *Primitive weighting function*, assigns the relevance score of a one-term document and a one-term query. It rewards the document $weight(q)$ when d matches q and penalize it by $penalty(q, d)$ otherwise. We assume that $\forall t \in T$, $weight(t) > 0$

and $\forall q, \forall d \neq q, penalty(q, d) < weight(d)$.

When it comes to longer documents and longer queries, we define a retrieval function inductively.

Inductive Step: $\forall Q, D$ such that $|Q| \geq 1$ and $|D| \geq 1$,
(1) Assume $Q' = Q \cup \{q\}$.

$$S(Q', D) = S(Q \cup \{q\}, D) = g(S(Q, D), S(\{q\}, D), q, Q, D)$$

(2) Assume $D' = D \cup \{d\}$.

$$S(Q, D') = S(Q, D \cup \{d\}) = h(S(Q, D), S(Q, \{d\}), d, Q, D)$$

Function g is called the *Query growth function*, and it describes the score change when one more term is added into a query. When a new term q is added, the score of any document for this new query (i.e. $S(Q \cup \{q\}, D)$) is determined by the score of the document for the old query (i.e. $S(Q, D)$), the score of the document for the added query term (i.e. $S(\{q\}, D)$), and any possible score adjustment through D, Q and q. Similarly, function h is called the *Document growth function*, and it describes the score change when adding one more term to a document. Similarly, the score of the new document for any query Q (i.e. $S(Q, D \cup \{d\})$) is related to the score of the old document (i.e. $S(Q, D)$), the score of the added term for the given query (i.e. $S(Q, \{d\})$, and certain additional information from D, Q and the added term d.

Functions f, g, and h define the framework of our search space. Unfortunately, above definitions themselves are too flexible to guarantee that every formula in the space is a reasonable function, because $S(Q, D)$ may be decomposed in multiple ways, and its values are likely decomposition-order-sensitive. To make the space meaningful, the following theorem gives a set of necessary and sufficient conditions under which $S(Q, D)$ can be guaranteed to be a function.

Theorem 1 *Let $\delta_d(d, D, Q) = S(Q, D \cup \{d\}) - S(Q, D)$ is the score change due to the addition of term d to document D, and $\delta q(q, D, Q) = S(Q \cup \{q\}, D) - S(Q, D)$ is the score change due to the addition of a term q to query Q. $S(Q, D)$ is a function if and only if all the following conditions holds.*
(1) $\forall Q, D$ and $\forall q, d \in T$, $\delta_d(d, D, Q) + \delta_q(q, D \cup \{d\}, Q) = \delta_q(q, D, Q) + \delta_d(d, D, Q \cup \{q\})$
(2) $\forall Q, D$ and $\forall d_1, d_2 \in T$, $\delta_d(d_1, D, Q) + \delta_d(d_2, D \cup \{d_1\}, Q) = \delta_d(d_2, D, Q) + \delta_d(d_1, D \cup \{d_2\}, Q)$
(3)$\forall Q, D$ and $\forall q_1, q_2 \in T$, $\delta_q(q_1, D, Q) + \delta_q(q_2, D, Q \cup \{q_1\}) = \delta_q(q_2, D, Q) + \delta_q(q_1, D, Q \cup \{q_2\})$

Intuitively, these three conditions simply require that $S(Q, D)$ remains the same no matter in which order the terms are added to the query and the document. The proof of this theorem is given as follow.

Proof *: **(Only if)** Suppose $S(Q, D)$ is a function. Therefore, the value of $S(Q, D)$ remains same no matter in which order the terms are added in each of the following three situations: (1) Add two different terms to the document; (2) Add two different terms to the query; (3) Add a term to the document and a term to the query.*

In the first situation, it requires that $\forall d_1, d_2$,

$$\begin{aligned} S(Q, D \cup \{d_1\} \cup \{d_2\}) &= S(Q, D \cup \{d_1\}) + \delta_d(d_2, D \cup \{d_1\}, Q) \\ &= S(Q, D \cup \{d_2\}) + \delta_d(d_1, D \cup \{d_2\}, Q). \end{aligned}$$

By further decomposing $S(Q, D \cup \{d_1\})$ and $S(Q, D \cup \{d_2\})$, we have

$$\delta_d(d_1, D, Q) + \delta_d(d_2, D \cup \{d_1\}, Q) = \delta_d(d_2, D, Q) + \delta_d(d_1, D \cup \{d_2\}, Q).$$

Similarly, for the second situation, it requires that $\forall q_1, q_2$,

$$\begin{aligned} S(Q \cup \{q_1\} \cup \{q_2\}, D) &= S(Q \cup \{q_1\}, D) + \delta_q(q_2, D, Q \cup \{q_1\}) \\ &= S(Q \cup \{q_2\}, D) + \delta_q(q_1, D, Q \cup \{q_2\}). \end{aligned}$$

By further decomposition, we have

$$\delta_q(q_1, D, Q) + \delta_q(q_2, D, Q \cup \{q_1\}) = \delta_q(q_2, D, Q) + \delta_q(q_1, D, Q \cup \{q_2\}).$$

Finally, for the third situation, it requires that $\forall d, q$,

$$\begin{aligned} S(Q \cup \{q\}, D \cup \{d\}) &= S(Q \cup \{q\}, D) + \delta_d(d, D, Q \cup \{q\}) \\ &= S(Q, D \cup \{d\}) + \delta_q(q, D \cup \{d\}, Q) \end{aligned}$$

After further decomposition, we have

$$\delta_q(q, D, Q) + \delta_d(d, D, Q \cup \{q\}) = \delta_d(d, D, Q) + \delta_q(q, D \cup \{d\}, Q).$$

(If) *Suppose those three questions hold. We want to prove that* $S(Q, D)$ *is a function for all* Q *and* D*, where* $|Q| \geq 1$ *and* $|D| \geq 1$*. We will prove it inductively.*

Base case: *Given* $|Q| = 1$ *and* $|D| = 1$*. Based on the definition of primitive weighting function, it is trivial that there is only one possible value of* $S(Q, D)$ *for any given* Q *and* D*, which means* $S(Q, D)$ *is a function.*

Inductive Step: *Given that* $S(Q', D')$ *is a function when* $2 \leq |D'| + |Q'| \leq k$ $(k \geq 2)$*, we need to show that* $S(Q, D)$ *is a function when* $|Q| + |D| = k + 1$*.*

First, based on one of the given equations, we have $\forall q_1, q_2 \in Q$, $|Q| \geq 2$,

$$\delta_q(q_1, D, Q - \{q_1\}) + \delta_q(q_2, D, Q') = \delta_q(q_2, D, Q - \{q_2\}) + \delta_q(q_1, D, Q'),$$

where $Q' = Q - \{q_1\} - \{q_2\}$*. Since* $|Q - \{q_1\} - \{q_2\}| + |D| \leq k$*,* $S(Q - \{q_1\} - \{q_2\}, D)$ *is a function. After adding it to both sides of the above equation, we have*

$$\delta_q(q_1, D, Q - \{q_1\}) + S(Q - \{q_1\}, D) = \delta_q(q_2, D, Q - \{q_2\}) + S(Q - \{q_2\}, D),$$

which demonstrates that the value of $S(Q, D)$ *remains same no matter which query term is the last one to be added.*

Second, based on the given equations, we have $\forall d_1, d_2 \in D$*,* $|D| \geq 2$*,*

$$\delta_d(d_1, D - \{d_1\}, Q) + \delta_d(d_2, D', Q) = \delta_d(d_2, D - \{d_2\}, Q) + \delta_d(d_1, D', Q),$$

where $D' = D - \{d_1\} - \{d_2\}$*. Since* $|Q| + |D - \{d_1\} - \{d_2\}| \leq k$*,* $S(Q, D - \{d_1\} - \{d_2\})$ *is a function. After adding it to both sides of the above equation, we have*

$$\delta_d(d_1, D - \{d_1\}, Q) + S(Q, D - \{d_1\}) = \delta_d(d_2, D - \{d_2\}, Q) + S(Q, D - \{d_2\})$$

which demonstrates that the value of $S(Q, D)$ *remains same no matter which term is the last one to be added to the document.*

Finally, we need to guarantee that the value of $S(Q, D)$ *remains same no matter whether the last term is added to document or to the query, which can be proved as follow. Based on the given equations, we have* $\forall d \in D, q \in Q$*,* $|D| \geq 2, |Q| \geq 2$*,*

$$\delta_d(d, D - \{d\}, Q) + \delta_q(q, D - \{d\}, Q - \{q\}) = \delta_q(q, D, Q - \{q\}) + \delta_d(d, D - \{d\}, Q - \{q\}).$$

Since $|Q - \{q\}| + |D - \{d\}| \leq k$*,* $S(Q - \{q\}, D - \{d\})$ *is a function. After adding it to both sides of equation, we have*

$$\delta_d(d, D - \{d\}, Q) + S(D - \{d\}, Q) = \delta_q(q, D, Q - \{q\}) + S(D - \{d\}, Q).$$

Therefore, if the above equations hold, the value of $S(Q, D)$ remains same no matter in which order the term is added to the document or/and query, i.e. $S(Q, D)$ is a function.

6.2 Inductive Function Decomposition

In order to obtain some sense about the relationship between the existing retrieval functions and this new way of defining a retrieval function, we rewrite 3 representative existing retrieval functions using the inductive definition schema.

6.2.1 Pivoted Normalization (PN)

After rewriting, we have

$$\begin{aligned}
weight(q) &= ln\frac{N+1}{df(q)} \cdot \frac{1}{Piv\_LN(1)} \\
penalty() &= 0 \\
g() &= S(Q, D) + S(\{q\}, D) \\
h() &= \lambda_1(|D|) \cdot S(Q, D) + \lambda_2(|D|) \cdot \Delta TF(C(d, D)) \cdot S(Q, \{d\})
\end{aligned}$$

where $Piv\_LN(x) = (1-s) + s\frac{x}{avdl}$, $\Delta TF(x) = ln(1 + ln(x+1)) - ln(1 + ln(x))$, $\lambda_1(x) = \frac{Piv\_LN(x)}{Piv\_LN(x+1)}$ and $\lambda_2(x) = \frac{Piv\_LN(1)}{Piv\_LN(x+1)}$.

The decomposition shows that $weight(q)$ is related to an IDF-related discriminative value of q, while $h()$ appears to implement document length and TF normalization.

6.2.2 Okapi

After rewriting, we have

$$\begin{aligned}
weight(q) &= ln\frac{N - df(q) + 0.5}{df(q) + 0.5} \cdot TF\_LN(1, 1) \\
penalty() &= 0
\end{aligned}$$

$$
\begin{aligned}
g() &= S(Q,D) + \Delta QTF(C(q,Q)) \cdot S(\{q\},D) \\
h() &= S(Q,D) + S(Q;\{d\}) \cdot \Delta TF(C(d,D),|D|+1) \cdot \gamma \\
&\quad + \sum_{t \in D \cap Q} S(Q,\{t\}) \cdot \Delta LN(C(t,D),|D|) \cdot \gamma \\
&= \sum_{t \in D \cap Q - \{d\}} S(Q,\{t\}) \cdot TF\_LN(C(t,D),|D|+1) \cdot \gamma \\
&\quad + S(Q;\{d\}) \cdot TF\_LN(C(d,D)+1,|D|+1) \cdot \gamma
\end{aligned}
$$

where $QTF(x) = \frac{(k_3+1)\times x}{k_3+x}$, $TF\_LN(x,y) = \frac{(k_1+1)\times x}{k_1((1-b)+b\frac{y}{avdl})+x}$,$\Delta TF(x,y) = TF\_LN(x+1,y) - TF\_LN(x,y)$, $\Delta LN(x,y) = TF\_LN(x,y+1) - TF\_LN(x,y)$, $\Delta QTF(x) = QTF(x+1) - QTF(x)$ and $\gamma = \frac{1}{TF\_LN(1,1)}$.

It shows again that $weight(q)$ is an IDF-related value of q. And $h()$ again implements length normalization and TF normalization, though the form of the formula is more complex than in the case of PN.

6.2.3 Dirichlet Prior (DP)

After rewriting, we have

$$
\begin{aligned}
weight(q) &= ln(1 + \frac{1}{\mu \cdot p(q|C)}) - ln(1 + \frac{1}{\mu}) \\
penalty() &= -ln(1 + \frac{1}{\mu}) \\
g() &= S(Q,D) + S(\{q\},D) \\
h() &= S(Q,D) + \beta(C(d,D),p(d|C)) \cdot S(Q,d) \\
&\quad + \theta(p(d|C),C(d,D),|D|,|Q|)
\end{aligned}
$$

where $\theta(x,y,z,l) = l \cdot (ln\frac{\mu}{z+1+\mu} - ln\frac{\mu}{z+\mu}) - \frac{ln(1+\frac{y+1}{x}) - ln(1+\frac{y}{x})}{ln(1+\frac{1}{x})} \cdot l \cdot ln\frac{\mu}{1+\mu}$ and $\beta(x,z) =$

$\frac{ln(1+\frac{x+1}{\mu\cdot z})-ln(1+\frac{x}{\mu\cdot z})}{ln(1+\frac{1}{\mu\cdot z})}$.

The results show that $weight(q)$ is yet again an IDF-related value of q. However, $penalty()$ is not equal to 0 as in the previous two methods; instead, it is a negative value, which also contributes document length normalization. Function h takes yet another complex form, involving not only TF and length normalization but also the IDF-like variable $p(d|C)$. Function θ is playing a role for additional score adjustment due to the addition of the terms.

6.2.4 Summary

The rewriting exercise provides interesting insights on how we may derive new functions.

1. All the instantiations of $weight(q)$ are related to an IDF-like discrimination value of q. However, $weight(q)$ in Okapi can be smaller than $penalty(q,d)(=0)$, which causes poor performance on verbose queries.

2. There are two ways to implement document length normalization in our framework. One is to set $penalty(w,q) < 0$, which penalizes any non-query terms in the document, as in the DP method. The other is to use document length related parameters to adjust the document relevance score as in PN (i.e. $\lambda_1()$ and $\lambda_2()$) and Okapi (i.e. $TF\_LN()$).

3. It shows three possible ways to instantiate the document growth function, which we summarize below in a more general form.

$$
\begin{aligned}
S(Q, D\cup\{d\}) &= \lambda_1(|D|)\cdot S(Q,D) + \lambda_2(|D|)\cdot \alpha(C(d,D))\cdot S(Q,\{d\}) \\
S(Q, D\cup\{d\}) &= \sum_{t\in D\cap Q-\{d\}} S(Q,\{t\})\lambda(|D|+1, C(t,D)) \\
&\quad +S(Q,\{d\})\cdot\lambda(|D|+1, C(d,D)+1) \\
S(Q, D\cup\{d\}) &= S(Q,D) + \beta(C(d,D), C(d,Q), p(d|C))\cdot S(Q,\{d\}) \\
&\quad +\theta(C(d,D), p(d|C), |D|, |Q|)
\end{aligned}
$$

6.3 Derivation of New Retrieval Functions

In this section, we study how to instantiate each component function in the framework to derive a new reasonable retrieval function. Our approach is to start with decomposition results of any existing retrieval function, generalize each component function of the existing functions, use constraints to find alternative implementation for each component function, and finally combine the three component functions to form a new retrieval function.

6.3.1 Primitive Weighting Function

The primitive weighting function has two component functions: $weight(q)$ and $penalty(q, d)$. As discussed in the previous section, the decision on $penalty(q, d)$ affects the instantiation of the document growth function, so we postpone its discussion together with the document growth function.

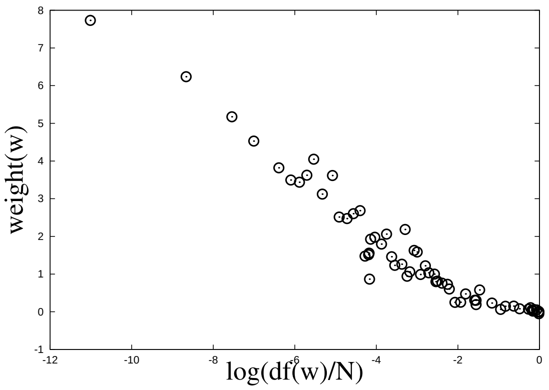
We consider two ways to define $weight(q)$, both connected with how the matching of q contributes to the relevance between documents and queries. The first is to define it as the point-wise mutual information between the presence/absence of q in a document ($p(occ)$) and whether the document is relevant to the given query ($p(rel)$).

$$weight(q) = log\frac{p(occ \cap rel)}{p(occ)p(rel)} = log\frac{p(occ|rel)}{p(occ)} \tag{6.2}$$

The second is to define it as the conditional probability that a document is relevant if q occurs in the document:

$$weight(q) = P(rel|occ) \tag{6.3}$$

$p(occ)$ can be estimated as $p(occ) = \frac{df(q)}{N}$. If the relevance information of documents is available (e.g. through feedback from the users), it is easy to estimate $p(occ|rel)$ and $p(rel|occ)$, so $weight(q)$ can be computed accordingly. However, when we have insufficient relevance information, it is much harder to compute $weight(q)$ directly. One possible solution is to employ exploratory data analysis [18, 20]. The basic idea is to find an empirical function that can well explain the relationship between unknown and known variables on training data. For example, we may relate $weight(q)$ to the known variables $p(occ)$ and try to find a function of $p(occ)$ that can approximate $weight(q)$ well. Specifically, for a given data set, we compute $weight(q)$ (according to Equation (6.2) or (6.3)) and $p(occ)$ for each query term. Since the variance of these variables is large, we follow [18] and group the data points together in bins. We average both known and unknown variables for a bin to obtain a "pseudo data point", and plot the graph for these two variables (i.e. $weight(q)$ vs. $p(occ)$).



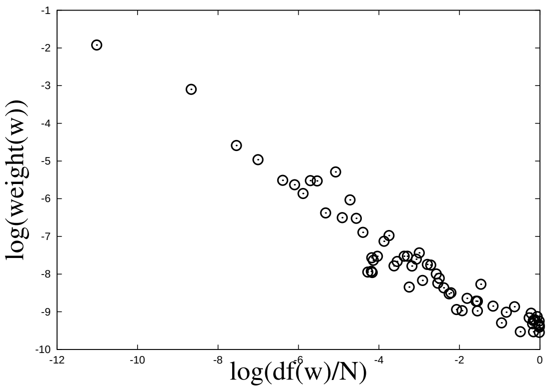


Figure 6.1: Plot of $weight(q)$(computed using Equation 6.2) vs. $log\frac{df(q)}{N}$ (Left) and plot of $log(weight(q))$ (computed using Equation 6.3) vs. $log\frac{df(q)}{N}$ (Right)

The left plot of Figure 6.1 shows the $weight(q)$ computed using Equation(6.2) against $log(P(occ)) = log\frac{df(q)}{N}$ on AP data set. (The plots on other data sets are similar.) There appears to be a negative linear correlation between them. Thus we assume $weight(q) = a \log \frac{df(q)}{N} + b$, where a and b are constants. Visually examining several such plots on different data sets indicates $a = -1$ and $b = 0$ may be a good approximation. That is $weight(q) = log\frac{N}{df(q)}$, which will be referred to as *LOG weighting function.* Note that the LOG weighting function is the typical IDF [61, 49].

The right plot in Figure 6.1 shows how $\log(weight(q))$, where $weight(q)$ is computed using Equation(6.3), is related to $log\frac{df(q)}{N}$. We also see a negative linear correlation between them. Again, as a crude approximation, we may assume

$$weight(q) = (\frac{N}{df(q)})^k, (0 < k < 1)$$

where k is a parameter. We call this formula *EXP weighting function.*

6.3.2 Query Growth Function

The analysis of existing retrieval functions reveals that their query growth functions are similar and of a relatively simple form. The slightly more complicated form of Okapi has not shown any clear benefit in our preliminary experiments. Thus we fix our choice of query growth function to the following simple form:

$S(Q \cup \{q\}, D) = S(Q, D) + S(\{q\}, D)$.

6.3.3 Document Growth Function

We generalize the document growth functions of the three existing retrieval functions and explore the interesting alternative choices.

Formula 1—PN Variation The generalized form of the PN document growth function is

$$S(Q, D \cup \{d\}) = \lambda_1(|D|) \cdot S(Q, D) + \lambda_2(|D|) \cdot \alpha(C(d, D)) \cdot S(Q, \{d\})$$

which is a weighted linear combination of $S(Q, D)$and $S(Q, \{d\})$ with the weights depending on three unknown functions (i.e. $\lambda_1, \lambda_2, \alpha$).

We can easily recover PN with the following instantiations:

$$\begin{aligned}\lambda_1(x) &= \frac{avdl \cdot \frac{1-s}{s} + x}{avdl \cdot \frac{1-s}{s} + x + 1}, \lambda_2(x) = \frac{avdl \cdot \frac{1-s}{s} + 1}{avdl \cdot \frac{1-s}{s} + x + 1} \\ \alpha(y) &= ln(1 + ln(y+1)) - ln(1 + ln(y)), (y > 1)\end{aligned}$$

We now discuss how we may exploit our inductive definition scheme and retrieval constraints to find interesting alternative instantiations of λ_1, λ_2 and α.

First, we need to ensure that $S(Q, D)$ is a function. Applying theorem 1, we find that condition (1) and condition (3) can be satisfied unconditionally, but the following two equations must hold in order to satisfy condition (2).

$$\lambda_2(k+1) = \lambda_1(k+1) \times \lambda_2(k), k \geq 0 \tag{6.4}$$

$$\lambda_2(0) = 1 \tag{6.5}$$

Next, TFC1 suggests that

$$\frac{\lambda_2(k)}{1 - \lambda_1(k)} > \frac{S(Q, D)}{weight(q)}. \tag{6.6}$$

Since $S(Q, D)$ is roughly a sum of weights over all matched terms, we expect $\frac{S(Q,D)}{weight(q)} < avdl$. Thus we consider the following stronger but simpler condition; if Equation (6.7) holds, we expect Equation (6.6) to hold for most documents.

$$\frac{\lambda_2(k)}{1 - \lambda_1(k)} > avdl. \tag{6.7}$$

Furthermore, LNC1 implies that

$$\forall k, \lambda_1(k) < 1. \tag{6.8}$$

One way to satisfy this condition is to let $\lambda_1(k) = \frac{f(k)}{f(k+1)}$, where $f(k)$ decreases when k increases. A natural simple choice for $f(k)$ is $f(k) = a \times k + b$, where $a > 0$. In this case, $\lambda_1(k) = \frac{a \times k + b}{a \times (k+1) + b}$. According to Equation (6.4), $\lambda_2(k) = \frac{f(1)}{f(k+1)} = \frac{a+b}{a \times (k+1) + b}, k > 0$. Therefore, Equation (6.7) is equivalent to $1 + \frac{b}{a} > avdl$. We thus can assume $\frac{b}{a} = avdl/s$ and $0 < s < 1$, and have

$$\lambda_1(k) = \frac{k + \frac{avdl}{s}}{k + 1 + \frac{avdl}{s}}, \lambda_2(k) = \frac{1 + \frac{avdl}{s}}{k + 1 + \frac{avdl}{s}}.$$

Finally, the analysis of TFC2 and TFC3 requires that $\alpha(C(d, D))$ decreases when $C(d, D)$ increases. It is easy to show that $\alpha(0) = 1$. So $\forall x, \alpha(x) \leq 1$. Leaving the study of a better form of $\alpha(C(d, D))$ for our future work, we simply take the corresponding component from the pivoted normalization formula: $\alpha(k) = ln(1 + ln(k + 1)) - ln(1 + ln(k)), k \geq 1$ and $\alpha(0) = 1$.

Using this document growth function together with $penalty(q, d) = 0$, we obtain the following retrieval function

$$S(Q, D) = \sum_{t \in D \cap Q} TF(C(t, D)) \cdot C(t, Q) \cdot weight(t) \frac{avdl + s}{avdl + |D| \cdot s} \tag{6.9}$$

where $0 \leq s \leq 1$ and $TF(x) = 1 + ln(1 + ln(x))$.

If we set $s = \frac{s'}{1-s'}$ and $weight(q) = log\frac{N+1}{df(q)}$, Equation 6.9 turns into PN with parameter s'. The constraint $0 \leq s \leq 1$ is equivalent to $0 \leq s' \leq 0.5$, which is a narrower range than the full range $(0, 1)$ allowed by the standard PN method. Empirical study [11] shows that the optimal value of s' is always smaller than 0.4, thus the new formula we derived using the axiomatic framework has a more reasonable parameter range than the original PN, due to the introduction of the extra constraint Equations (6.6) and (6.7).

Formula 2—Okapi Variation The generalized form of the Okapi document growth function is

$$S(Q, D \cup \{d\}) = \sum_{t \in D \cap Q - \{d\}} S(Q, \{t\}) \lambda(|D| + 1, C(t, D)) + S(Q, \{d\}) \cdot \lambda(|D| + 1, C(d, D) + 1)$$

It differs from the document growth function of PN in that the linear combination weights are related to not only the document length but also the term count and we only have one unknown function (i.e. λ) to instantiate The following instantiation clearly recovers Okapi. $\lambda(x, y) = \frac{(k_1+1) \times y}{k_1((1-b)+b\frac{x}{avdl})+y}$.

We now explore how to find any interesting alternative instantiations of $\lambda(x, y)$, where x is related to the document length and y is related to the term count. We check all the constraints to see whether they can provide us more clues about λ.

All the three conditions in Theorem 1 are satisfied unconditionally. TFC1 indicates that $\lambda(x + 1, y + 1) > \lambda(x, y)$, and LNC1 shows that $\lambda(x + 1, y) < \lambda(x, y)$, which means $\lambda(x, y)$ decreases when x increases. From these, it follows that $\lambda(x + 1, y + 1) > \lambda(x + 1, y)$, i.e., $\lambda(x, y)$ increases as y increases. TFC2 and TFC3 indicate that $\lambda(x, y)$ should be a sublinear function w.r.t. y. From the Okapi instantiation, $\lambda(x, y)$ controls how to penalize a long document as well as how to normalize the term frequency for every term. We consider a slightly more general form of the Okapi instantiation, $\lambda(x, y) = \frac{y}{(ax+b)+y}$. The analysis of TFC1 implies that $\frac{b}{a} > avdl \times r, (0 < r < 1)$ and $0 < b \leq 1$. One way to satisfy this condition is to set $a = s/avdl$ and $b = s$, where $0 < s \leq 1$. Using this document growth function together with $penalty(q, d) = 0$, we obtain

$$S(Q, D) = \sum_{t \in D \cap Q} C(t, Q) \cdot weight(t) \cdot \frac{C(t, D)}{\frac{s}{avdl} \cdot |D| + s + C(t, D)}$$

Formula 3—DP Variation Different from PN and Okapi, the DP method partially implements length normalization through setting a negative value to $penalty(q, d)$. The generalized form of the DP document growth function is

$$\begin{aligned} penalty(d,q) &< 0 \\ S(Q, D \cup \{d\}) &= S(Q,D) + \beta(C(d,D), C(d,Q), p(d|C)) \cdot S(Q,d) \\ &\quad + \theta(p(d|C), C(d,D), |D|, |Q|) \end{aligned}$$

Setting $penalty() = -ln(1 + \frac{1}{\mu})$ and setting β and θ as follows will recover the DP function.

$$\begin{aligned} \beta(x,y,z) &= \frac{ln(1+\frac{x+1}{\mu \cdot z}) - ln(1+\frac{x}{\mu \cdot z})}{ln(1+\frac{1}{\mu \cdot z})} \\ \theta(x,y,z,l) &= l \cdot (ln\frac{z+\mu}{z+1+\mu} - \frac{ln(1+\frac{y+1}{x}) - ln(1+\frac{y}{x})}{ln(1+\frac{1}{x})} \cdot ln\frac{\mu}{1+\mu}) \end{aligned}$$

To seek for interesting alternative instantiations, we follow DP and set $penalty(q,d) = c$ where c is a negative constant. We consider a simple case where $\theta() = 0$ and $\beta()$ is only related to $C(d,D)$ and $C(d,Q)$ as follows.

$$\beta(C(d,D), C(d,Q), p(d|C)) = \beta'(C(d,D), C(d,Q)) = \begin{cases} \alpha(C(d,D)), & C(d,Q) \neq 0 \\ 1, & C(d,Q) = 0 \end{cases}$$

$\beta'(C(d,D), C(d,Q))$ captures the change of term frequency. When d is a query term (i.e. $C(d,Q) > 0$), the change of term frequency is captured by $\alpha(C(d,D))$. The function α can be constrained by TFC2 and TFC3. We use the same implementation of $\alpha()$ in PN. On the contrary, when d is a non-query term (i.e. $C(d,Q) = 0$), we simply assume that the score change due to the addition of d is always the same. To balance the score between the reward and the penalty, we assume $c = -s/avdl$, where $0 \leq s \leq 1$. We obtain the following hybrid variation of PN and DP:

$$S(Q,D) = \sum_{t \in Q \cap D} C(t,Q) \cdot weight(t) \cdot TF(C(t,D)) - \gamma(|D|,|Q|)$$

where $TF(x) = 1 + ln(1 + ln(x)), \gamma(x,y) = \frac{(x-y)\cdot y \cdot s}{avdl}$ and $0 \leq s \leq 1$.

6.3.4 Derived Retrieval Functions

Combining all the choices, we obtain the following six new retrieval functions.

F1-LOG(s): $S(Q,D) = \sum_{t \in Q \cap D} C(t,Q) \cdot TF(C(t,D)) \cdot LN(|D|) \cdot LW(t)$

F1-EXP(s,k): $S(Q,D) = \sum_{t \in Q \cap D} C(t,Q) \cdot TF(C(t,D)) \cdot LN(|D|) \cdot EW(t)$

F2-LOG(s): $S(Q,D) = \sum_{t \in Q \cap D} C(t,Q) \cdot TF\_LN(C(t,D),|D|) \cdot LW(t)$

F2-EXP(s,k): $S(Q,D) = \sum_{t \in Q \cap D} C(t,Q) \cdot TF\_LN(C(t,D),|D|) \cdot EW(t)$

F3-LOG(s) $S(Q,D) = \sum_{t \in Q \cap D} C(t,Q) \cdot TF(C(t,D)) \cdot LW(t) - \gamma(|D|,|Q|)$

F3-EXP(s,k) $S(Q,D) = \sum_{t \in Q \cap D} C(t,Q) \cdot TF(C(t,D)) \cdot EW(t) - \gamma(|D|,|Q|)$ where $TF(x) = 1 + ln(1 + ln(x))$, $LW(t) = ln\frac{N+1}{df(t)}$, $EW(t) = (\frac{N+1}{df(t)})^k$, $LN(x) = \frac{avdl+s}{avdl+x\cdot s}$, $TF\_LN(x,y) = \frac{x}{x+s+\frac{s\cdot y}{avdl}}$ and $\gamma(x,y) = \frac{(x-y)\cdot y\cdot s}{avdl}$, $0 \leq s \leq 1$ and $0 \leq k \leq 1$.

In fact, F2-EXP is a re-parameterization of Okapi function with only one parameter. Previous studies [49, 79] have also attempted to vary components to form various retrieval formulas in an arbitrary way. Our framework provides more guidance on how to choose the components and can guarantee the performance of the derived functions in the sense of constraint satisfaction.

6.4 Experiments

In this section, we experimentally compare the derived new retrieval functions with the three existing ones. We also examine their parameter sensitivity. Our experiment results show

Table 6.1: Optimal Performance Comparison of the Derived Formulas

Formula	**Trec7**				**Trec8**			
	sk	sv	lk	lv	sk	sv	lk	lv
F1-LOG(Piv)	0.176	0.146	——-	0.199	0.245	0.205	——-	0.234
F1-EXP	0.184	0.173	——-	0.211	0.243	0.225	——-	0.251
F2-LOG	0.185	0.159	——-	0.208	0.260	0.210	——-	0.240
F2-EXP	0.187	0.186	——-	0.225	0.257	0.236	——-	0.260
F3-LOG	0.180	0.154	——-	0.204	0.244	0.206	——-	0.240
F3-EXP	0.187	0.180	——-	0.213	0.244	0.227	——-	0.250

Formula	**Web**				**FR**			
	sk	sv	lk	lv	sk	sv	lk	lv
F1-LOG(Piv)	0.288	0.212	——-	0.214	0.225	0.143	0.269	0.208
F1-EXP	0.288	0.228	——-	0.241	0.223	0.144	0.267	0.200
F2-LOG	0.295	0.245	——-	0.266	0.223	0.164	0.271	0.241
F2-EXP	0.289	0.272	——-	0.292	0.222	0.169	0.268	0.241
F3-LOG	0.290	0.213	——-	0.213	0.223	0.141	0.265	0.203
F3-EXP	0.288	0.229	——-	0.235	0.218	0.142	0.265	0.191

Formula	**AP**				**DOE**			
	sk	sv	lk	lv	sk	sv	lk	lv
F1-LOG(Piv)	0.226	0.193	0.385	0.292	0.179	0.105	0.269	0.210
F1-EXP	0.223	0.197	0.376	0.278	0.172	0.119	0.269	0.207
F2-LOG	0.227	0.201	0.386	0.296	0.184	0.110	0.270	0.209
F2-EXP	0.225	0.203	0.379	0.280	0.175	0.116	0.269	0.203
F3-LOG	0.227	0.192	0.386	0.295	0.180	0.103	0.266	0.212
F3-EXP	0.225	0.196	0.377	0.272	0.173	0.111	0.271	0.203

Table 6.2: Performance Comparison with Existing Formulas—Top 25-percentile

Formula	**Trec7**				**Trec8**			
	sk	sv	lk	lv	sk	sv	lk	lv
F2-EXP	**0.187**	**0.187**	——-	0.224	**0.256**	**0.236**	——-	**0.260**
F2-EXP-0.5	0.186	0.186	——-	**0.225**	0.250	**0.236**	——-	**0.260**
Pivoted	0.174	0.145	——-	0.196	0.239	0.201	——-	0.230
Okapi	0.185	0.084	——-	0.073	0.251	0.101	——-	0.108
Mod-Okapi	0.185	0.159	——-	0.215	0.252	0.218	——-	0.253
Dirichlet	0.186	0.182	——-	0.224	0.251	0.228	——-	0.259

Formula	**Web**				**FR**			
	sk	sv	lk	lv	sk	sv	lk	lv
F2-EXP	0.289	**0.272**	——-	**0.291**	0.223	**0.167**	0.267	0.234
F2-EXP-0.5	0.282	**0.272**	——-	**0.291**	0.222	0.164	0.266	0.227
Pivoted	0.253	0.207	——-	0.212	0.207	0.139	0.250	0.207
Okapi	0.310	0.203	——-	0.229	**0.229**	0.080	**0.276**	0.079
Mod-Okapi	**0.312**	0.244	——-	0.279	0.226	0.162	0.274	**0.251**
Dirichlet	0.289	**0.272**	——-	**0.291**	0.206	0.157	0.244	0.233

Formula	**AP**				**DOE**			
	sk	sv	lk	lv	sk	sv	lk	lv
F2-EXP	0.223	0.197	0.377	0.276	0.175	0.114	0.268	0.203
F2-EXP-0.5	0.220	0.190	0.374	0.272	0.174	0.112	0.268	0.203
Pivoted	0.225	0.190	0.383	0.288	0.179	0.102	0.263	0.206
Okapi	**0.226**	0.082	**0.385**	0.025	**0.184**	0.081	0.265	0.072
Mod-Okapi	**0.226**	0.194	0.384	**0.295**	0.183	0.104	0.270	0.216
Dirichlet	0.224	**0.204**	0.375	0.292	0.181	**0.125**	**0.276**	**0.228**

Table 6.3: Performance Comparison with Existing Formulas—Bottom 25-percentile

Formula	**Trec7**				**Trec8**			
	sk	sv	lk	lv	sk	sv	lk	lv
F2-EXP	**0.183**	**0.177**	——-	**0.212**	**0.243**	**0.214**	——-	**0.241**
Pivoted	0.053	0.048	——-	0.077	0.085	0.083	——-	0.095
Okapi	0.161	0.059	——-	0.053	0.223	0.085	——-	0.088
Mod-Okapi	0.165	0.135	——-	0.161	0.227	0.171	——-	0.185
Dirichlet	0.175	0.154	——-	0.202	0.235	0.209	——-	0.240

Formula	**Web**				**FR**			
	sk	sv	lk	lv	sk	sv	lk	lv
F2-EXP	0.275	**0.253**	——-	**0.254**	**0.216**	**0.150**	**0.258**	**0.205**
Pivoted	0.041	0.042	——-	0.051	0.061	0.056	0.082	0.070
Okapi	0.215	0.139	——-	0.153	0.186	0.054	0.228	0.058
Mod-Okapi	0.223	0.219	——-	0.197	0.199	0.132	0.251	0.171
Dirichlet	**0.282**	0.233	——-	0.234	0.189	0.138	0.202	0.171

Formula	**AP**				**DOE**			
	sk	sv	lk	lv	sk	sv	lk	lv
F2-EXP	0.206	0.160	0.351	0.237	**0.171**	0.100	0.252	0.187
Pivoted	0.089	0.072	0.184	0.135	0.067	0.039	0.122	0.090
Okapi	0.208	0.076	0.371	0.023	0.167	0.062	0.250	0.051
Mod-Okapi	0.211	**0.178**	**0.375**	**0.270**	0.170	0.095	0.257	0.186
Dirichlet	**0.212**	**0.178**	0.357	0.267	0.159	**0.114**	**0.259**	**0.207**

Table 6.4: Performance Comparison with Existing Formulas—Average variance

Formula	**Trec7**				**Trec8**			
	sk	sv	lk	lv	sk	sv	lk	lv
F2-EXP	**5.6e-06**	**2.2e-05**	——-	**3.7e-05**	**3.8e-05**	**1.1e-04**	——-	**6.9e-05**
Pivoted	2.5e-03	1.7e-03	——-	2.4e-03	4.1e-03	2.4e-03	——-	3.1e-03
Mod-Okapi	7.0e-05	1.6e-04	——-	7.3e-04	1.1e-04	4.7e-04	——-	9.8e-04
Dirichlet	1.9e-05	2.8e-04	——-	1.6e-04	4.5e-05	1.5e-04	——-	6.9e-05

Form.	**Web**				**FR**			
	sk	sv	lk	lv	sk	sv	lk	lv
F2-EXP	**3.4e-05**	**7.3e-05**	——-	**2.7e-04**	**9.2e-06**	**5.7e-05**	**1.5e-05**	**1.4e-04**
Pivoted	7.4e-03	4.6e-03	——-	4.4e-03	3.6e-03	1.2e-03	4.6e-03	3.3e-03
Mod-Okapi	1.4e-03	1.3e-04	——-	1.4e-03	1.4e-04	2.2e-04	8.4e-05	1.3e-03
Dirichlet	1.3e-04	5.2e-04	——-	6.9e-04	6.0e-05	9.5e-05	3.8e-04	9.2e-04

Form.	**AP**				**DOE**			
	sk	sv	lk	lv	sk	sv	lk	lv
F2-EXP	4.6e-05	2.3e-04	1.2e-04	2.6e-04	**4.1e-06**	3.2e-05	5.2e-05	**5.3e-05**
Pivoted	3.2e-03	2.5e-03	7.2e-03	4.2e-03	2.3e-03	7.1e-04	3.7e-03	2.3e-03
Mod-Okapi	4.9e-05	4.9e-05	**1.6e-05**	**1.3e-04**	3.3e-05	**1.3e-05**	**2.9e-05**	1.8e-04
Dirichlet	**2.7e-05**	**2.2e-04**	5.8e-05	2.4e-04	7.3e-05	3.9e-05	5.3e-05	7.9e-05

that the new functions can generally achieve comparable optimal performance with the three existing functions, but are more robust over different data collections and less sensitive to the parameter settings. We set k in the EXP weighting function to 0.35 based on our preliminary experiments.

6.4.1 Comparison of Derived Functions

To compare the optimal performances of the six derived functions, we vary the parameter value from 0 to 1.0 and select a best run with the highest average precision for each function on each data set. We compare the average precisions of these best runs in Table 6.1.

We make the following observations. First, the optimal performances of all the six functions are comparable, and F1-LOG (i.e., PN) is relatively worse than others. Second, the functions with EXP weighting usually perform better than those with LOG weighting for verbose queries, but worse for keyword queries. Finally, the functions with F2 usually perform better than those with F1 and F3. The parameter sensitivity study also shows that F2-EXP appears to be more stable than others. So, it appears that F2-EXP is overall a better choice than others. Below we compare its performance with existing retrieval functions.

6.4.2 Comparison with Existing Functions

We compare the performance of one derived formula (i.e.F2-EXP) with PN, Okapi, and DP. Due to the poor performance of the original Okapi on verbose queries, we also compare with the modified Okapi (i.e., Okapi with traditional IDF) [11]. Since other retrieval functions all have one parameter, we set $k_1 = 1.2$, $k_3 = 1000$ [] and only vary the value of b in Okapi and modified Okapi. So, the Okapi performance reported in the thesis is not fully tuned. For every method, we randomly sample 12 values within the range of the parameter. Skewed samples with 25% or more of the values falling into an interval of 0.1 are discarded. For each method on each collection, we select the top/bottom 25-percentile runs (i.e., 3 runs with the best/worst average precision) from the 12 runs for comparison.

The results are shown in Table 6.2 (top 25-percentile) and Table 6.3 (bottom 25-percentile). F2-EXP0.5 is F2-EXP with a fixed value of 0.5 for s. From Table 6.2, we see that the optimal performance of F2-EXP is quite comparable with that of all the existing retrieval formulas. Even the fixed parameter value F2-EXP0.5 is also comparable, demonstrating the robustness of this axiomatic retrieval function.

The robustness is further confirmed in Table 6.3, where we see that F2-EXP mostly outperforms others and in Table 6.4, where we see that the average variance of all 12 runs for F2-EXP is mostly smaller than for all others.

It is interesting to note that modified Okapi always performs better than F2-EXP on AP; indeed, AP and DOE seem to be the only data sets where F2-EXP has not shown advantages. Further analysis and experiments are clearly necessary to better understand this.

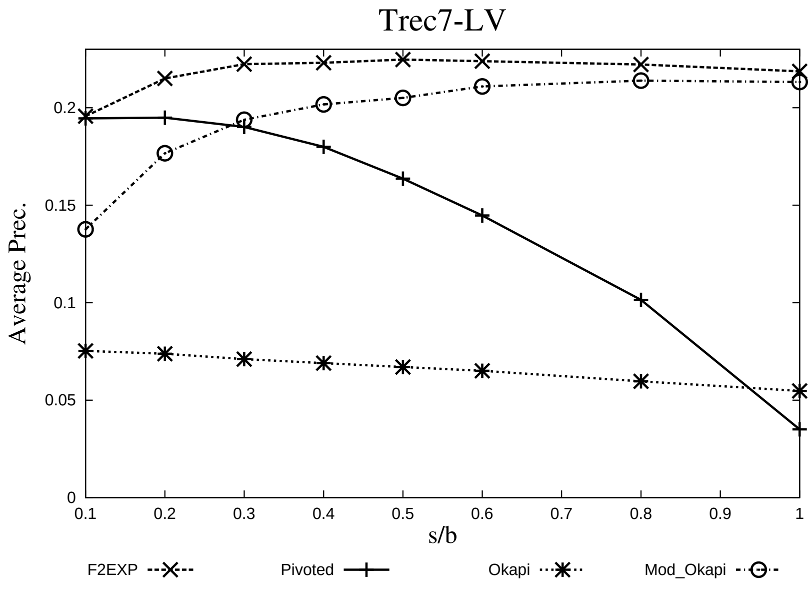


Figure 6.2: Performance Sensitivity on Trec7-LV

6.4.3 Parameter Sensitivity

We compare the parameter sensitivity between F2-EXP, PN Okapi, and modified Okapi. We did not include DP because its parameter is in a different scale, but it is known that its performance is sensitive to the smoothing parameter [77]. We vary the parameter from 0 to 1. The results on TREC7 are shown in Figure 6.2. The plot demonstrates the stability of F2-EXP, which we have also observed in the plots for other data sets and query types.

6.5 Conclusions and Future Work

In this chapter, we discuss how to derive new retrieval functions in the axiomatic framework, in which the notion of relevance is directly captured by retrieval constraints. The basic idea is to search for a retrieval function that satisfies all retrieval constraints. In order to search the function space more efficiently, we propose an inductive scheme for retrieval function definitions.

The inductive definition scheme provides a common basis to analytically compare different retrieval functions. We compare and analyze three representative existing retrieval functions in our framework and find that while the three functions implement similar heuristics, they implement them in different ways.

We further derive new retrieval functions using the axiomatic framework. We use both intuitive retrieval constraints and exploratory data analysis to guide us in instantiating the three components of the inductive definition and obtaining several new retrieval functions. We evaluate these new retrieval functions on a number of representative test sets. The experiment results show that the derived new functions are more stable than the existing retrieval functions with comparable optimal performance.

As discussed in this chapter, the axiomatic framework allows us to derive more robust and effective retrieval functions in a principled way. This thesis has only moved a very small step in this direction. There are many interesting future research directions.

First, we have used only basic constraints when deriving new retrieval functions. Presumably, with more reasonable constraints, the derived functions will be more specialized and performing better. It would be interesting to define additional constraints and incorporate them to a retrieval function in a systematical way. In the next chapter, we discuss how to incorporate semantic term matching to further improve the performance, which is one step in this important direction.

Second, it would be interesting to study how to define the function space in a more general way. The current way of function space decomposition has some limitations. For example, such inductive definition forces us to search in a subspace of retrieval function instead of in the space including all possible retrieval functions. Furthermore, such inductive definition prevents us from capturing the whole query concept, because the global information of the query is not available with the inductive definition. Thus, it would be interesting to study some other way to decompose the function space so that it would allow us to efficiently search through the function space without introducing unnecessary limitations. A possible solution is to construct a search space based on different variables. Although existing retrieval functions are motivated very differently, they indeed share several common variables, such as the term count, the document frequency and the document length. These variables can serve as a starting point. With the assumption that the relevance score is additive, we define a retrieval function as follows.

$$S(Q,D) = \sum_{t \in Q \cup D} \tau(c(t,D), c(t,Q), weight(t,Q), |D|, |Q|),$$

where $c(t,D)$ is the term count of t in a document D, $c(t,Q)$ is the term count of t in a query Q, $weight(t,Q)$ is the function to identify the importance of a term t for the given query Q, and $|D|$ $|Q|$ are the document length and the query length respectively.

We can instantiate the function τ as follows.

$$\begin{aligned}
S(Q,D) &= \sum_{t \in Q \cup D} \tau(c(t,D), c(t,Q), weight(t,Q), |D|, |Q|) \\
&= \sum_{t \in Q \cap D} \tau_{11}(c(t,D), |D|, weight(t,Q)) \times \tau_{12}(c(t,Q)) \\
&\quad + \sum_{t \in \overline{Q} \cap D} \tau_2(|D|, |Q|, c(t,D), weight(t,Q)) \\
&\quad + \sum_{t \in Q \cap \overline{D}} \tau_3(|Q|, c(t,Q), |D|, weight(t,Q)),
\end{aligned}$$

In this way, τ is decomposed to several components, where τ_{11} regulates the relevance score change when a query term t is added to the document, τ_{12} captures the score change when a query term t is added to the query, τ_2 defines how to assign a score to the document when it contains a non-query term t and τ_3 captures the score change when a query term t does not appear in the document D. Indeed, such instantiation is general enough to cover most existing retrieval functions without any unnecessary assumptions. We hope that such decomposition would allow us to find more efficient and robust retrieval functions.

Third, we are interested in designing a mechanism to systematically search for a retrieval function in the whole function space, instead of searching in only the neighborhood of existing functions as we did in this thesis.

Chapter 7

Incorporating Semantic Term Matching

The axiomatic framework provides a natural way to incorporate additional information to improve the performance by introducing more retrieval constraints. In this chapter, we study how to incorporate semantic term matching. A common limitation of many retrieval models is that relevance scores are solely based on *exact* (i.e., syntactic) matching of terms in the queries and documents, without allowing distinct but semantically related terms to match each other and contribute to the retrieval score. In the axiomatic framework, we show that semantic term matching can be naturally incorporated into the axiomatic retrieval model through defining the primitive weighting function based on a semantic similarity function of terms. We define several desirable retrieval constraints for semantic term matching and use such constraints to extend the axiomatic model to directly support semantic term matching based on the mutual information of terms computed on some document set. We show that such extension can be efficiently implemented as query expansion. Experiment results on several representative data sets show that, with mutual information computed over the documents in either the target collection for retrieval or an external collection such as the Web, our semantic expansion consistently and substantially improves retrieval accuracy over the baseline axiomatic retrieval model. As a pseudo feedback method, our method also outperforms a state-of-the-art language modeling feedback method.

7.1 Semantic Term Matching Constraints

Let $s(t, u) \in [0, +\infty]$ be any given semantic similarity function between two terms t and u. Without loss of generality, we assume that term t is semantically more similar to term u than to term v if and only if $s(t, u) > s(t, v)$, i.e., a large value of s indicates a high similarity. Since intuitively a term has the highest similarity to itself, we assume $\forall u \neq t, s(t, t) > s(t, u)$. We also assume that s is symmetric, i.e., $\forall t, u, s(t, u) = s(u, t)$. Based on such a semantic similarity function, we now define three constraints that we would like any reasonable retrieval function to satisfy.

STMC1: Let $Q = \{q\}$ be a query with only one term q. Let $D_1 = \{d_1\}$ and $D_2 = \{d_2\}$ be two single-term documents, where $q \neq d_1$ and $q \neq d_2$. If $s(q, d_1) > s(q, d_2)$, then $S(Q, D_1) > S(Q, D_2)$.

STMC1 requires a retrieval function to give a higher score to a document with a term that is more semantically related to a query term. Thus, even though D_1 and D_2 do not match the query Q syntactically, we would like D_1 to have a higher score because term d_1 is more semantically related to query term q than term d_2 is. Clearly, STMC1 directly constrains the primitive weighting function.

STMC2: Let $Q = \{q\}$ be a single term query and d be a non-query term such that $s(q, d) > 0$. If D_1 and D_2 are two documents such that $|D_1| = 1$, $c(q, D_1) = 1$, $|D_2| = k$ and $c(d, D_2) = k$ ($k \geq 1$), where $c(q, D_1)$ and $c(d, D_2)$ are the counts of q and d in D_1 and D_2 respectively, then $S(Q, D_1) \geq S(Q, D_2)$.

STMC2 requires that matching an original query term q exactly should always contribute no less to the relevance score than matching a semantically related term d, no matter how many times term d occurs in the document.

STMC3: Let $Q = \{q_1, q_2\}$ be a query with only two query terms and d be a non-query term such that $s(q_2, d) > 0$. Let D_1 and D_2 be two documents. If $|D_1| = |D_2| > 1$, $S(\{q_1\}, \{q_1\}) = S(\{q_2\}, \{q_2\})$, $c(q_1, D_1) = |D_1|$, $c(q_1, D_2) = |D_2| - 1$ and $c(d, D_2) = 1$, then

$S(Q, D_1) \leq S(Q, D_2)$.

STMC3 intends to capture the following intuition: Suppose we have a query with two *equally* important terms q_1 and q_2. Suppose a document D_1 matches q_1 n (> 1) times, but does not match q_2 or any of its semantically related terms. If we change one of the occurrences of q_1 in D_1 to a term semantically related to q_2 to form a document D_2, D_1 should not have a lower score than D_2, because D_2 covers more distinct query terms than D_1.

7.2 Extension Based on STMCs

The constraints defined above provide some guidance on how to extend the inductively defined axiomatic retrieval functions to incorporate semantic term matching.

First, it is clear that these existing axiomatic functions violate all the three constraints we defined, simply because the semantic similarity function s is not part of the retrieval function. For example, based on the primitive weighting function defined in previous section, any single-term document will be assigned a zero score if the term in the document is not matching exactly the query term, which clearly violates STMC1.

To make the primitive weighting function satisfy STMC1, a natural solution is to define the following *generalized primitive weighting function* based on a given similarity function s.

$$S(\{q\}, \{d\}) = \omega(q) \times f(s(q, d)),$$

where f is a monotonically increasing function. Note that it is reasonable to require $\forall q \in Q, f(s(q, q)) = 1$ for any query Q, because the score of generalized primitive weighting function should be comparable with the score of the original one when the two terms match exactly. One way to ensure such property is to define f in terms of normalized similarity.

$$f(s(q,d)) = \frac{s(q,d)}{s(q,q)} \times \lambda(q,d)$$

where

$$\lambda(q,d) = \begin{cases} 1 & q = d \\ \beta & q \neq d \end{cases} \tag{7.1}$$

β is used to regulate the weighting of the original query terms and the semantically similar terms. The value of β should satisfy $0 < \beta < \frac{s(q,q)}{s(q,d)}$, because $f(s(q,d)) < f(s(q,q))$ $= 1)$ when $d \neq q$.

The generalized primitive weighting function clearly satisfies STMC1, and if we combine it with any existing instantiations of document growth function and query growth function, the derived retrieval functions would also satisfy STMC1 unconditionally. We further analyze STMC2 and STMC3 on such derived functions and find that these constraints are satisfied when β is within a certain range. Specifically, the analysis of STMC2 provides a tighter upper bound for β, while the analysis of STMC3 provides a tighter lower bound. The actual values of these bounds depend on the instantiation of document growth function. As an example, the lower and upper bounds for F2-EXP is:

$$\frac{b}{2+b} \times \frac{s(q,q)}{s(q,d)} \leq \beta \leq \frac{1}{b+1} \times \frac{s(q,q)}{s(q,d)} \tag{7.2}$$

We see that the bounds of β depend on both the query and semantic similarity function s. In our experiments, on each data set, the lower bound of β is determined by the lowest value of $\frac{s(q,q)}{s(q,d)}$ for all the query terms while the upper bound of β is determined by the highest value of $\frac{s(q,q)}{s(q,d)}$, which are the minimal requirements of β.

Since a term can potentially have a huge number of semantically related terms, the computation of the generalized retrieval functions can be expensive. To reduce the computation cost, we can reasonably restrict our attention to the most similar terms for each query term. Such simplification is not expected to affect the retrieval score significantly, because the dropped terms would contribute little to the score anyway. Thus we redefine the generalized primitive weighting function as follows:

$$S_{gen}(\{q\},\{d\}) = \begin{cases} \omega(q) \cdot \frac{s(q,d)}{s(q,q)} \cdot \lambda(q,d) & d \in \varepsilon(q) \\ 0 & otherwise \end{cases} \tag{7.3}$$

where $\varepsilon(q)$ is the set of K most semantically similar terms of q according to the similarity function s, $\omega(q)$ is as in Equation (6.1) and $\lambda(q,d)$ is defined in Equation(7.1).

Even with this simplification, the computation can still potentially involve enumerating all the combinations of query terms and document terms. Fortunately, there is an efficient way to compute such a retrieval function based on query expansion as shown in the next section.

7.3 As Query Expansion

Let us first introduce some notations. $S(Q,D)$ is the scoring function of the original inductively defined axiomatic retrieval function, where only syntactic term matching is considered. $S_{gen}(Q,D)$ is the generalized inductively defined axiomatic retrieval function obtained by combining the generalized primitive weighting function with the original document growth and query growth function.

The generalized primitive weighting function (i.e., Equation (7.3)) can be re-written as follows.

$$S_{gen}(\{q\},\{d\}) = \begin{cases} \omega(q) & d = q \\ \omega(q:d) & d \in \varepsilon(q)/\{q\} \\ 0 & otherwise \end{cases}$$

where $\omega(q:d) = \omega(q) \times \beta \times \frac{s(q,d)}{s(q,q)}$.

Let $\varepsilon'(q)$ be the set of K most semantically similar terms of q excluding itself, i.e., $\varepsilon'(q) = \varepsilon(q)/\{q\}$. Let P be the set of the K most similar terms of all query terms, i.e., $P = \bigcup_{q\in Q}(\varepsilon'(q))$. $\forall t \in P$, let $\rho(t)$ be the set of query terms that are semantically similar to t. Define S' such that $\forall t \in P$,$S'(\{t\},\{t\}) = \omega(\rho(t):t) = \frac{\sum_{u\in\rho(t)}\omega(u:t)}{|Q|}$; otherwise $S'(Q,D) = S(Q,D)$.

Theorem: $\forall Q, D$, $S_{gen}(Q,D) = S'(Q \cup P, D)$.

Proof:

$$\begin{aligned} S_{gen}(Q,D) &= \sum_{q\in Q} S_{gen}(q,D) \\ &= \sum_{q\in Q}(S_{gen}(q,D_q) + \sum_{t\in\varepsilon'(q)\cap D} S_{gen}(t,D_t)) \\ &= \sum_{q\in Q}(S(q,D_q) + \sum_{t\in\varepsilon'(q)\cap D} S'(t,D_t)) \\ &= S(Q,D) + \sum_{q\in Q}\sum_{t\in\varepsilon'(q)\cap D} S'(t,D_t) \\ &= S(Q,D) + \sum_{t\in P} S'(t,D_t) \\ &= S'(Q,D) + \sum_{t\in P} S'(t,D) \\ &= S'(Q\cup P, D) \end{aligned}$$

where D_t is the part of the document D that only contains t. (i.e., $|D_t| = c(t,D) = c(t,D_t)$).

The first step is based on query growth function. The second step assumes that the relevance score of a document can be computed as the sum of the disjoint subsets of the document, which holds for all the inductively defined axiomatic retrieval functions. The third step is based on the fact that S_{gen} and S' use the same document growth function and the fact that $S'(\{t\}, \{t\}) = \omega(\rho(t) : t)$ is consistent with generalized primitive function when $t \in P$.

The theorem shows that scoring a document using S_{gen} can be reduced to scoring using S' with an expanded query formed by adding, for each query term, K most similar terms to the query. Note that the weight of a similar term t is computed from $\omega(\rho(t) : t)$ instead of $\omega(t)$ as used in the traditional query expansion methods.

7.4 Term Semantic Similarity Function

The remaining challenge is to define $s(t_1, t_2)$ in STMC1. In general, we may exploit any knowledge and resources available to us to compute term similarity and there are many ways to compute it. For example, co-occurrences of terms obtained from the analysis of a document collection usually reflect underlying semantic relationships that exist between terms [54, 5, 7, 4], and we may use measures such as Dice similarity [1] and mutual information [66, 34, 22, 17, 32, 16] to compute term similarity. In this thesis, we adopt the mutual information as the basic semantic similarity metric, leaving other choices for future work.

The mutual information (MI) of two terms t and u in a set of documents can be computed as follows [66]:

$$I(X_t, X_u) = \sum_{X_t, X_u \in \{0,1\}} p(X_t, X_u) \log \frac{p(X_t, X_u)}{p(X_t)p(X_u)}$$

X_t and X_u are two binary random variables corresponding to the presence/absence of term t and term u in each document or segment.

Mutual information is a principled way to measure term correlations, and it satisfies our requirements about the similarity function s. The next choice we have to make is which corpus to use when computing the mutual information. A natural choice would be the document collection from which we retrieve documents. However, such a choice may not be ideal because an ambiguous term can have multiple senses in a large corpus. As a result, the semantically related terms found by mutual information could be a mix of terms corresponding to different senses of the original term, introducing noise in query expansion. Thus, it is crucial to compute mutual information over a "clean" corpus, where ideally only one (correct) sense of the query term occurs. How can we find such a "clean" corpus? One possibility is to use the top-M documents returned by the retrieval systems for the query. The rational is that we can reasonably assume there is only one sense of a query term in the set of relevant documents, and the top-M documents are reasonable approximations of the set of relevant documents. This is indeed in line with what previous work in query expansion has found – local document analysis tends to be more effective than global document analysis [75].

However, the top-M documents would clearly be a *biased* corpus, and in this sense, it is not a good corpus for computing mutual information. For example, it is likely that a query term occurs in all the top-M documents. The abundance of a query term would then cause popular terms in the top-M documents to generally have a high mutual information. In particular, a common term (e.g., "can") would have a high mutual information, even if it also occurs in many other documents where the query term does not occur. To solve this problem, we need to supplement the top-M documents with some additional documents that do not necessarily contain any query term. Thus we will randomly choose $r \times M$ documents from the collection and combine them with the top-M documents as a mixed corpus for computing mutual information.

Clearly, the choice of r may also affect the mutual information results. How do we choose a good value for r? Once again, constraint analysis can provide some guidance. The following notations will be used in defining the constraints: N is the total number of documents in the document collection. $df(t)$ is the number of documents that contain t in the collection. W is the working set containing $r \times M$ random documents plus the top M documents returned by the system; since the $r \times M$ documents are chosen from the documents ranked below the top-M documents, we clearly have $M + M \times r \leq N$. $df(t_1, t_2|W)$ is the number of documents that contain both t_1 and t_2 in the working set W. $df(t|W)$ is the number of documents that contain t in the working set W.

Intuitively, the value of r should not be very small, because we need enough number of random documents to penalize the common terms. Consider the scenario in Figure 7.1(a), where t_1 is a "truly" semantically related term, while t_2 is a common term. t_1 is semantically more similar to q than t_2, although t_2 co-occurs with q in more documents than t_1. This intuition can be captured by the following Term Semantic Similarity Constraint(TSSC).

TSSC1: Let q be a query term and t_1 and t_2 be two non-query terms. If $df(q, t_1|W) = \frac{M}{2}$, $df(t_1|W) = \frac{M}{2}$, $df(q|W) = M$, $df(q, t_2|W) = M$, $df(t_2|W) = M + \frac{r \times M}{2}$, then $s(q, t_1) > s(q, t_2)$.

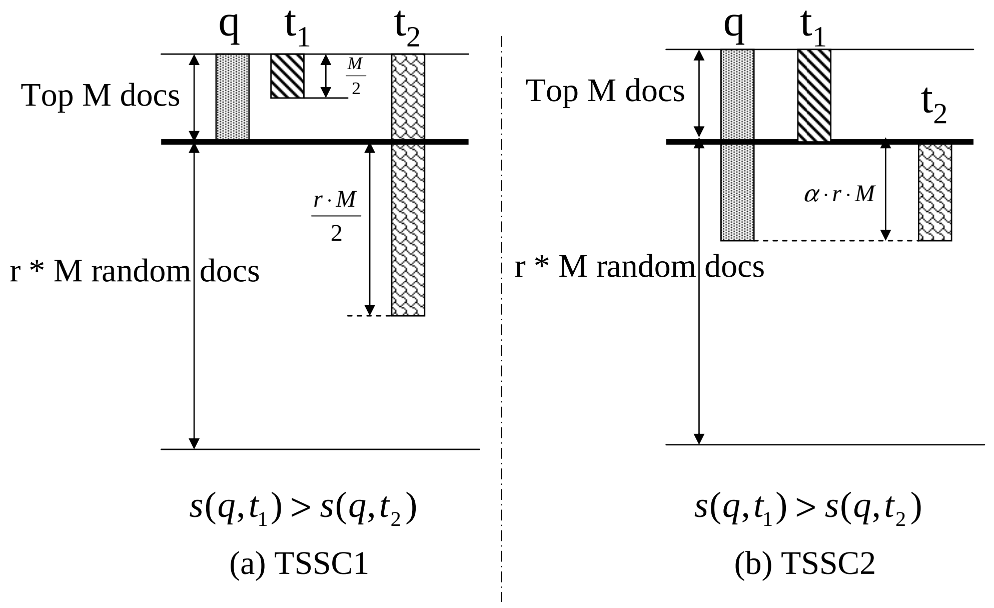


Figure 7.1: TSSC

On the other hand, the value of r should not be very large because we want to ensure that the dominant sense of a query term is the one determined by the whole query. Consider the scenario in Figure 7.1(b). Suppose a query term q has two senses. The first sense is the one determined by the whole query (i.e., in the top M documents), and a term t_1 is semantically related to this sense of q (i.e., they co-occur in the top M documents). Now suppose another term t_2 is semantically related to another sense of q (i.e., they co-occur in the random documents). Intuitively, t_1 should have a higher similarity score than t_2. The following constraint captures this intuition.

TSSC2: Let q be a query term and t_1 and t_2 be two non-query terms. If $0 < \alpha < 1$, $df(q, t_1|W) = M$, $df(t_1|W) = M$, $df(q|W) = M + \alpha \times r \times M$ and $df(t_2|W) = df(q, t_2|W) = \alpha \times r \times M$, then $s(q, t_1) > s(q, t_2)$.

α is the percentage of the documents that contain q in a random sample of the whole collection after the top M documents excluded, i.e., $\alpha = \frac{df(q)-M}{N-M}$.

The above two constraints are satisfied only when the value of r is within a certain range. Indeed, TSSCs provide a lower and an upper bounds for r.

$$1 < r < \frac{N}{df(q)} \tag{7.4}$$

The value of r is collection and query dependent. For each collection, we use the median of the document frequency of all query terms to compute the upper bound of r.

7.5 Summary

We briefly summarize the high-level steps involved in the proposed method for incorporating semantic term matching:

1. Construct a working set where term semantic similarity can be computed.
2. For every query term, find the top L most similar terms based on the working set.
3. Gather the top L similar terms for all the query terms, then select the top K ranked

terms based on
$\omega(\rho(t) : t)$.

4. Expand the original query with the K terms. Note that the weight of an expansion term is computed based on $\omega(\rho(t) : t)$ instead of $\omega(t)$.

In the first step, the working set can be constructed over any reasonable resources in the following way: Given any collection of documents and a query, we first use the original inductively defined axiomatic retrieval function to rank the documents. We then merge the top M returned documents with $r \times M$ random documents selected from the same collection to form a working set for computing term similarity. The collection to be used can be either the target collection for retrieval (called *internal expansion*) or any other collections (called *external expansion*). To form a large pool of terms, L is usually fixed to 1000. Four parameters need to be tuned: the number of expansion terms (i.e., K), the number of top documents (i.e., M), the number of random documents (i.e., r) and the scaling parameter β. The optimal values of β and r are expected to be within a certain range based on Equation (7.2) and Equation(7.4), which is also supported by our experiment results.

7.6 Experiments

7.6.1 Experiment Design

We conduct three sets of experiments. First, we evaluate the effectiveness of the semantic term matching. Second, we examine the parameter sensitivity of the method. Finally, we compare it with a model-based feedback method in language modeling approaches [76].

All experiments are conducted over two collections used in recent Robust track [71, 72]: (1) TREC Disk 4&5 (minus Congressional Record) with 249 official topics of Robust track in 2004. The document set has 1908MB text and 528,000 documents. This is labeled as “ROBUST04”. (2)AQUAINT data with 50 official topics of Robust track in 2005. The document set has 3GB text and 1,033,461 documents. This is labeled as "ROBUST05". Some experiments are also conducted over six other data sets used in the previous sections. In all the experiments, we use the title-only queries, because short keyword query is the most frequently used query type by web users and semantic term matching is necessary for such short queries.

The performance is measured using the official measures in Robust track: MAP (mean average precision) and gMAP (geometric mean average precision). gMAP [72, 71] is a variant of the traditional MAP measure that uses a geometric mean rather than an arithmetic mean. This measure emphasizes the performance of poorly-performing topics.

The preprocessing only involves stemming with Porter's stemmer. As pointed out in the previous work [12], using a fixed parameter value ($b = 0.5$), F2-EXP can often achieve near-optimal performance in many test sets. Thus, we fix b to 0.5 in our experiments. We use the optimal value of b for the other five inductively defined axiomatic retrieval functions. In the first and third sets of experiments, M and K are both fixed to 20 and r is fixed to 29, so that we will get a total of 600 documents in the working set. We tune the value of β and report the best performance unless otherwise stated. **BL** is the baseline method without expansion (i.e., without semantic term matching). **docAX** and **segAX** are semantic expansion methods with MI computed based on co-occurrences in documents and 100-word segments, respectively. In all the result tables, ‡ and † indicate that the improvement is statistically significant according to Wilcoxon signed rank test at the level of 0.05 and 0.1 respectively.

7.6.2 Effectiveness of Semantic Term Matching

Table 7.1 shows the performance of the internal expansion for all six functions. The semantic term matching consistently and significantly outperforms the baseline on both data sets in terms of MAP. But, gMAP decreases in a few cases, which indicates that most of the performance improvement comes from the easy topics. F2-EXP is the best of all the functions.

We further test the semantic expansion on top of F2-EXP on six other data sets, and found that semantic expansion outperforms the baseline (i.e., F2-EXP) significantly (Table 7.2) on all the data sets except FR88-89. Due to the limit of space, we only report the performance of F2-EXP in the remaining experiments.

Table 7.3 shows the performance when semantic similarity is computed over the internal resource (i.e., collection itself), the external resource (i.e., a pool of Google snippets returned for a query), and both (i.e., first use external expansion, then do another round of internal expansion). We make the following observations. First, the expansion method improves the performance significantly in all cases. Second, the web-based external expansion method is consistently more effective than the internal expansion method in both measures. This indicates that the use of good external resources improves the effectiveness especially over the poorly-performing topics, which is consistent with what others have observed [72]. Finally, combining both internal and external expansion further improves the accuracy.

7.6.3 Sensitivity Analysis

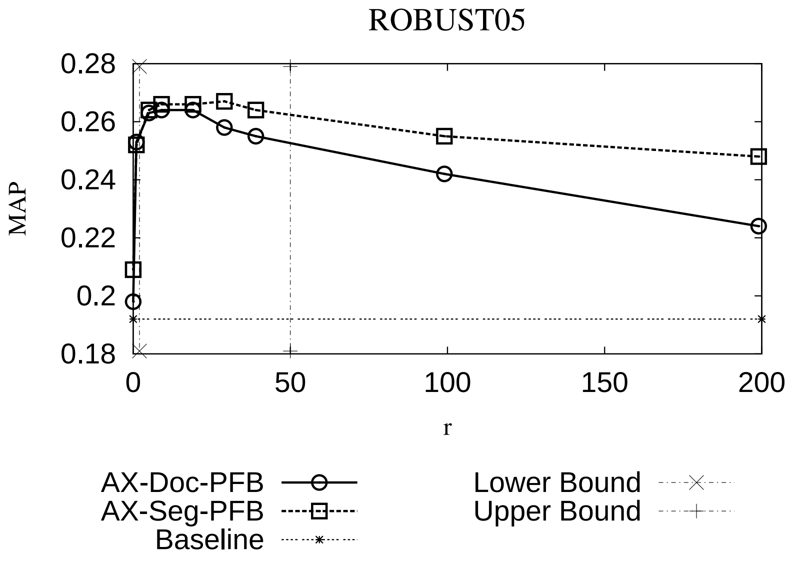


Figure 7.2: Performance Sensitivity (r)

Table 7.1: Performance of different axiomatic functions.

Method		ROBUST04		ROBUST05	
		MAP	gMAP	MAP	gMAP
F1-LOG	BL	0.241	0.138	0.200	0.131
	docAX	0.261 **8.3%‡**	0.150 **8.7%‡**	0.241 **21%‡**	0.126 **-3.8%‡**
	segAX	0.267 **11%‡**	0.148 **7.3%‡**	0.256 **28%‡**	0.134 **2.3%‡**
F1-EXP	BL	0.240	0.137	0.199	0.128
	docAX	0.262 **9.2%‡**	0.150 **9.5%‡**	0.246 **24%‡**	0.126 **-2.4%‡**
	segAX	0.266 **11%‡**	0.148 **8.0%‡**	0.252 **27%‡**	0.128 **0.0%**
F2-LOG	BL	0.251	0.141	0.196	0.125
	docAX	0.278 **11%‡**	0.157 **11%‡**	0.270 **38%‡**	0.131 **4.8%‡**
	segAX	0.284 **13%‡**	0.156 **11%‡**	0.281 **43%‡**	0.135 **8.0%‡**
F2-EXP	BL	0.248	0.142	0.192	0.122
	docAX	0.285 **15%‡**	0.157 **11%‡**	0.258 **34%‡**	0.136 **11%‡**
	segAX	0.288 **16%‡**	0.158 **11%‡**	0.267 **39%‡**	0.137 **12%‡**
F3-LOG	BL	0.240	0.138	0.200	0.131
	docAX	0.259 **7.9%‡**	0.146 **5.8%‡**	0.241 **21%‡**	0.138 **5.3%‡**
	segAX	0.267 **11%‡**	0.149 **7.9%‡**	0.253 **27%‡**	0.131 **0.0%**
F3-EXP	BL	0.239	0.137	0.198	0.127
	docAX	0.261 **9.2%‡**	0.150 **9.5%‡**	0.244 **23%‡**	0.125 **-1.6%‡**
	segAX	0.265 **11%‡**	0.148 **8.0%‡**	0.254 **28%‡**	0.130 **2.4%‡**

Table 7.2: Performance of F2-EXP on more data sets.

Data	MAP			gMAP		
	BL	docAX	segAX	BL	docAX	segAX
TREC7	0.186	0.236 **27%‡**	0.247 **33%‡**	0.083	0.098 **18%‡**	0.098 **18%‡**
TREC8	0.250	0.277 **11%‡**	0.278 **11%‡**	0.147	0.172 **17%‡**	0.167 **14%‡**
WEB2g	0.282	0.324 **15%‡**	0.324 **15%‡**	0.188	0.220 **17%‡**	0.220 **17%‡**
FR88-89	0.217	0.227 **4.6%**	0.224 **3.2%**	0.058	0.062 **6.9%**	0.069 **19%**
AP88-89	0.220	0.266 **21%‡**	0.267 **21%‡**	0.074	0.088 **19%‡**	0.086 **16%‡**
DOE	0.174	0.186 **6.9%‡**	0.184 **5.8%‡**	0.069	0.078 **13%‡**	0.074 **7.3%‡**

Table 7.3: Performance when using different resources.

Method		ROBUST04		ROBUST05	
		MAP	gMAP	MAP	gMAP
BL		0.248	0.142	0.192	0.122
Internal Expansion	docAX	0.285 **15%‡**	0.157 **11%‡**	0.258 **34%‡**	0.136 **11%‡**
	segAX	0.288 **16%**	0.158 **11%‡**	0.267 **39%‡**	0.137 **12%‡**
External Expansion		0.300 **21%‡**	0.196 **38%‡**	0.270 **41%‡**	0.196 **61%‡**
External + Internal Expansion	docAX	0.300 **21%‡**	0.178 **25%‡**	0.289 **51%‡**	0.203 **66%‡**
	segAX	0.302 **22%‡**	0.175 **23%‡**	0.290 **51%‡**	0.198 **62%‡**

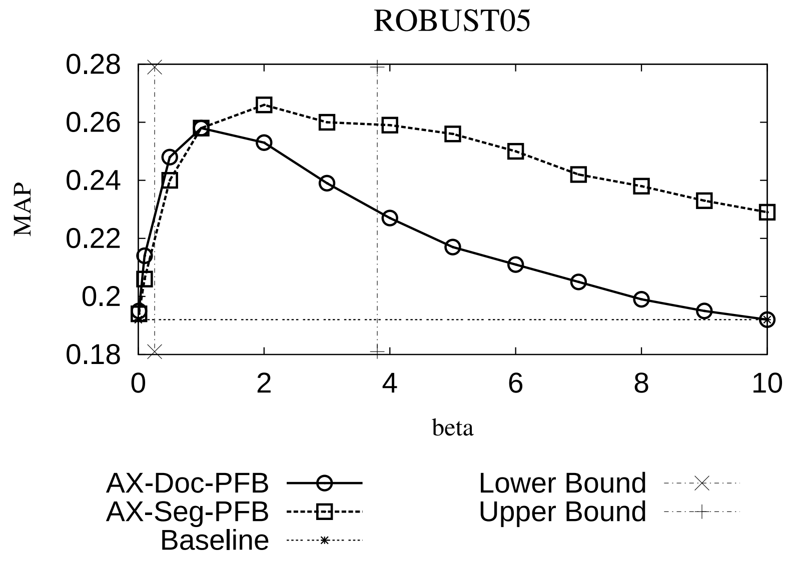


Figure 7.3: Performance Sensitivity (β)

Next, we study the performance sensitivity for the four parameters in the semantic expansion. Here we only show plots on ROBUST05, but similar trends can be observed for all the other data sets. Figure 7.2 shows the sensitivity curve for r. Equation (7.2) gives $1 < r < 50$ for the ROBUST05 data set. The performance is relatively stable when r is within the range, while it decreases when r is out of the range. Figure 7.3 shows the sensitivity curve for β. Equation (7.4) gives $0.27 \leq \beta \leq 3.8$ for the ROBUST05 data set. The optimal value is indeed within the predicted range, although docAX and segAX have different optimal value of β. Figure 7.4 shows the sensitivity curve for K. The performance is near optimal when K is 20. The performance is relatively stable when more terms are added. Figure 7.5 shows the curve for M. We observe the performance is optimal when M is around 20. The performance decreases when more documents are used, likely because the assumption that top M documents are all relevant is not true for larger values of M.

7.6.4 Comparison with Feedback Methods

Both our semantic expansion and traditional feedback methods select terms for query expansion. Traditional feedback methods [47, 76] select terms that have higher weight in the

Table 7.4: LM Feedback & Additive Effect

Method	ROBUST04		ROBUST05	
	MAP	gMAP	MAP	gMAP
BL	0.251	0.140	0.196	0.131
Internal PFB	0.275	0.139	0.254	0.105
IPFB + **docAX**	0.284	0.151	0.269	0.133
	3.3%‡	**8.6%**‡	**5.9%**†	**27%**†
IPFB + **segAX**	0.283	0.144	0.280	0.138
	2.9%†	**3.6%**†	**10%**‡	**31%**‡
External PFB	0.282	0.172	0.226	0.156
EPFB + **docAX**	0.293	0.170	0.278	0.168
	3.9%‡	**-1.2%**‡	**23%**†	**7.7%**†
EPFB + **segAX**	0.293	0.168	0.279	0.166
	3.9%‡	**-2.3%**‡	**24%**‡	**6.4%**‡

Table 7.5: Performance (MAP) of term selection (segAX)

Data	**Weighting Function**	**Term Selection**	
		KL-Div.	F2-EXP
ROBUST04	KL-Div.	0.275	0.288 (4.72%)‡
	F2-EXP	0.285	0.288 (1.05%)
ROBUST05	KL-Div.	0.254	0.273 (7.48%)‡
	F2-EXP	0.265	0.267 (0.755%)

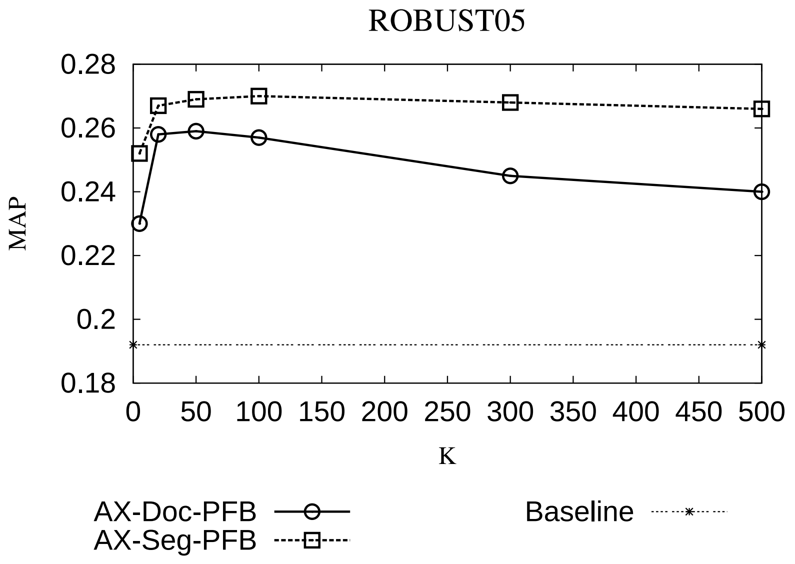


Figure 7.4: Performance Sensitivity (Num of Terms)

feedback documents, while our method selects terms that are semantically related to any query term. It would be interesting to compare their performance. In Table 7.4, we report the performance of the model-based feedback method in language modeling approaches [76]. Internal PFB (IPFB) is the pseudo feedback method. External PFB (EPFB) is the feedback method where the feedback terms are obtained from the Google snippets. We set μ to the optimal value for each data set, the number of feedback terms to 20 and the number of documents to 20. We tune the value of feedback coefficient and the value of mixture noise [76] and report the best performance. Comparing Tables 7.3 and 7.4 shows that the expansion method in axiomatic framework outperforms the model-based feedback method for both internal and external feedback. More interestingly, as shown in Table 7.4, our method can be combined with the traditional feedback methods to further improve performance, which shows that our method is complementary with the traditional feedback method.

Finally, we design experiments to study whether the performance gain of the semantic expansion comes from better term selection or from better term weighting. Assume A and B can be either our expansion method (i.e., F2-EXP) or traditional method (i.e., KL-Div.). We use method A to select terms for the method B, which means that we exclude any terms

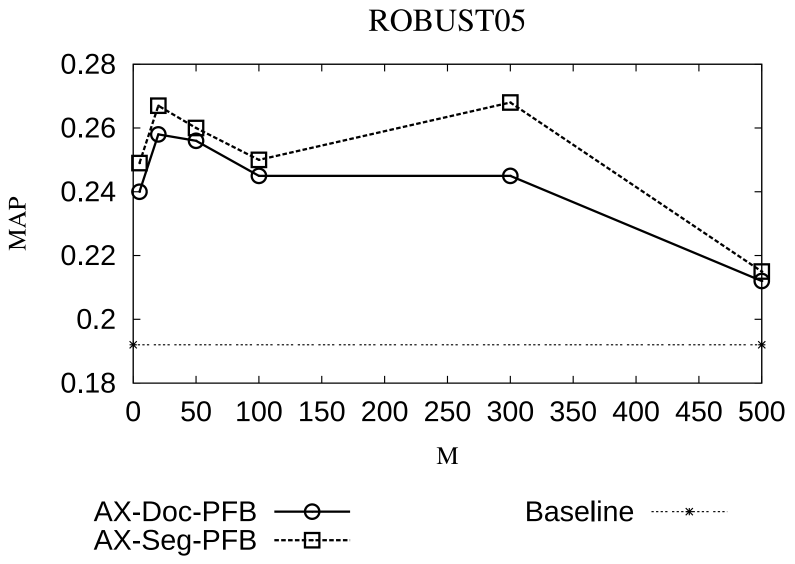


Figure 7.5: Performance Sensitivity (Num of Docs)

that are not nominated by A when using B. However, these terms are still weighted using B. This way we have four combinations shown in Table 7.5. The performance of using the terms selected by F2-EXP is consistently better than that of using the terms selected by KL-divergence method. For example, on ROBUST05 data set, the performance of KL-divergence method can be improved from 0.254 to 0.273 by using the terms selected by our method. The results indicate that the performance improvement of our method clearly comes more from better term selection.

7.7 Conclusions and Future Work

In this chapter, we propose a natural way to incorporate semantic term matching into axiomatic retrieval models. Several retrieval constraints are defined to capture intuitions on semantic term matching. The advantage of the axiomatic approach is that the constraints provide us guidance on the parameter setting and on the choice of term semantic similarity measure. Our proposed approach can be efficiently implemented as a query expansion method in the axiomatic framework.

The expansion based on semantic term matching was evaluated on several representative large retrieval collections. The results show that the proposed method is effective for all the six inductively defined axiomatic retrieval functions. Furthermore, the method works for both internal resources (e.g. collection itself) and external resources (e.g. the results returned by Google). The parameter sensitivity confirms the hypothesis that the constraint analysis can provide an upper bound and a lower bound for the optimal values of r and β. The performance is relatively stable when the values of the parameters are set within the range derived from the constraint analysis. As query expansion, the proposed method outperforms the model-based feedback method in language modeling approach and is shown to be complementary to the traditional feedback methods and can be combined with them to further improve performance.

There are many interesting future research directions. First, we can use more resources, such as WordNet [7, 69, 30, 33, 36], to compute term semantic similarity. So far we have only used co-occurrence information, such as mutual information, as a way to measure the semantic similarity between terms. It would be very interesting to use more resources, such as WordNet [7, 69, 30, 33, 36], to compute sophisticated term semantic similarity function to see whether we could achieve further performance improvement. Second, the proposed approach can also be applied to cross-lingual retrieval task. It would be interesting to see how well our method can perform in the cross-lingual retrieval task. Third, we only studied the expansion for title-only query. One interesting extension is to study the same problem for the verbose queries (such as description-only queries). One difficulty is that such term-based expansion method may introduce noises to the expanded terms. The possible solution is to define a few constraints based on the criterion of term selection. Finally, the term similarity between query terms are ignored in the thesis. It would be interesting to study the expansion based on query concepts instead of individual query term, which is along the line of [35, 39].

Chapter 8

Conclusions

In this chapter, we summarize our research findings and discuss the future research directions.

8.1 Summary

The study of retrieval models is fundamental for Information Retrieval. Developing robust and effective models is a long-standing challenge. Although retrieval models have been studied for decades, the current retrieval models are still far from satisfactory. In particular, there is no way to predict the performance of a retrieval function analytically. As a result, heavy parameter tuning is always necessary for any state of the art retrieval function to achieve optimal performance.

This thesis presents a novel axiomatic approach to information retrieval, which sheds light on how to solve this long-standing challenge. In the axiomatic framework, relevance is directly modeled with term based retrieval constraints, which can be regarded as a set of axioms for retrieval task. Our goal is to search for a retrieval function that satisfies all the axioms, which is referred to as axiomatic approach. The central problem in the axiomatic framework is to design a set of axioms, i.e., a set of retrieval constraints that any reasonable retrieval function should satisfy. To solve this problem, we first summarize the retrieval heuristics that are commonly used in effective retrieval functions, such as TF-IDF, and then formalize these heuristics as retrieval constraints. These retrieval constraints describe the desirable properties of a retrieval function, explicitly state the users' requirements for relevant documents, and serve as the foundation of the framework.

This thesis makes the contributions in the following three aspects.

Predict the empirical performance of a retrieval function analytically: We analyze three representative retrieval functions with the defined retrieval constraints, and find that satisfaction of retrieval constraints is closely related to the empirical performance of a retrieval function. In particular, when a retrieval function violates a constraint, it performs poorly. As a result, conditional satisfaction often leads to a reasonable bound for the parameter value in a retrieval function. Therefore, constraint analysis allow us to avoid relying on labor-intensive and time-consuming empirical evaluate in order to decide whether a retrieval function is effective and to set the reasonable values for parameters in retrieval functions.

Diagnose the weaknesses and strengths of retrieval functions: We propose two different ways to identify the weaknesses of a retrieval function so that we could know how to modify a retrieval function to further improve its empirical performance.

First, constraint analysis provides us a natural way to identify the problem of a retrieval function analytically. If a retrieval function violates a constraint, the function has poor implementation for the corresponding aspect that is associated with the constraint. We use Okapi function as an example, and show that the retrieval performance can be improved if we modify the retrieval function to make it satisfy the constraints that are originally violated.

Second, we propose a novel general evaluation methodology to empirically identify the potential problems of retrieval functions. Existing evaluation methodology is not informative enough to explain the performance difference among different retrieval functions. However, such knowledge is necessary for developing the optimal retrieval function. To address this challenge, we first formally define a set of relevance-preserved collection perturbation operators, which can enlarge the performance difference and make it easier to pinpoint the weaknesses and strengths of retrieval functions. These operators serve as basic tools for us to perform diagnostic tests. We then present a common procedure to design the diagnostic tests for retrieval models. Following the procedure, we design three sets of diagnostics tests, and perform the tests on six data sets. These diagnostic results allow us to identify the weaknesses and strengths of a retrieval function, to explain the empirical differences among retrieval functions, and to give hints on how a retrieval function should be modified to further improve the performance. Based on the identified weaknesses, we modify the existing retrieval functions to overcome these weaknesses. Empirical results show that the modified retrieval functions outperform the state-of-art retrieval functions in most cases.

Develop more robust and effective retrieval functions: The fundamental difference between the axiomatic framework and any existing retrieval framework is that relevance is modeled *directly* through retrieval constraints. The basic idea is to search for a retrieval function that satisfies all the constrains. The underlying assumption is that if a retrieval function satisfies all the retrieval constraints, the retrieval function performs well empirically. We propose an inductive definition for a retrieval function. Such inductive definition can decompose a retrieval function to three component functions. To search in such a function space, our current strategy is to start from an existing retrieval function. We then search in the neighborhood of the existing retrieval function, and try to find a retrieval function that can satisfy all retrieval constraints in the neighborhood. With the inductive definition, the problem of searching for a good retrieval function boils down to the problem of searching for good implementations of the three component functions, which allow us to search in the function space efficiently. Experiments show that the derived retrieval functions are less sensitive to the parameter setting than the existing retrieval functions with the comparable optimal performance.

The axiomatic framework provides a natural way to incorporate other useful information to further improve the retrieval performance by defining a set of additional desirable constraints for such information. We choose the problem of incorporating semantic term matching as a case study in the thesis. We first define several retrieval constraints on semantic term matching, and then extend the derived axiomatic retrieval functions to satisfy these new constraints. Empirical results confirm the effectiveness of our approach and the potential of axiomatic approach in the aspect of incorporating more useful information to further improve the retrieval performance.

8.2 Future Research Directions

The axiomatic framework opens up many new possibilities for exploring and developing principled retrieval models. We demonstrate the great potential of the framework in three major aspects: (1) It makes it possible to predict the performance of a retrieval function analytically. (2) It allows us to diagnose the weaknesses of retrieval functions to further improve the performance. (3) It provides a novel way to develop more robust and effective retrieval functions. Since relevance is directly modeled through retrieval constraints, the framework enables us to understand relevance theoretically and predict the performance of a retrieval function analytically. The framework facilitates diagnostic analysis of retrieval functions, thus can provide guidances on how to eventually develop the ultimate optimal retrieval function. Some specific future research directions include:

Formalize more retrieval constraints: It is always better to have more constraints. We expect that the derived retrieval function performs better when we have more constraints. Ideally, if we could enumerate all desirable retrieval constraints, the target retrieval function would be guaranteed to be optimal. Unfortunately, it is extremely difficult, if not impossible, to find all desirable constraints. However, even if we could not find the complete set of desirable constraints, such incomplete set of constraints are still useful for us to find a more robust and effective retrieval function. As demonstrated in this work, most of the existing retrieval function can not even satisfy the most intuitive retrieval constraints unconditionally, which tells us the existing retrieval function are far from optimal. If we could find some reasonable retrieval constraints that none of the existing retrieval functions can satisfy, it would be highly possible to derive a more robust and effective retrieval function based on these constraints. Such more robust and effective retrieval function could benefit any search engine to further improve their performance. On the contrary, when we introduce more and more constraints, it is possible to find one reasonable retrieval constraint that conflicts with the other one. In fact, this would be also interesting, because, in this case, such conflicting constraints might provide some theoretic justifications on the ceiling performance of the existing retrieval models, which is along a similar line to a related work on clustering algorithms [24]. Although the average effectiveness of along a similar line to a related work on clustering algorithms [24]. Although the average effectiveness of these retrieval models increases about 10 percent every year in the early TREC, their performance seems to have reached a plateau[2]. It is unclear that whether the current performance of the state-of-art retrieval function is indeed the best performance we can achieve and whether we can find a retrieval function which can perform better. We hope that conflicts between retrieval constraints could provide some justification for this phenomenon.

Achieve theoretic understanding of relevance: Many retrieval models have been proposed and studies, and they all try to model the relevance in some way. Unfortunately, there is no formal definition for topic relevance. Relevance information is usually defined as the information that satisfies a user's information need. It seems that relevance is a quite subjective notation. Is there any way to formally define such subjective definition? Hopefully, our axiomatic framework could help us to answer this question. The formalized retrieval constraints allow us to connect the relevance with term statistics directly. We hope that such term-level constraints could help us to find a formal definition for topic relevance, which is along the similar line as the previous on similarity [29].

Derive the optimal retrieval function: Developing an optimal retrieval function has huge impact, because it will improve the accuracy of every search engine. The proposed axiomatic framework has a great potential for developing new optimal retrieval functions, that are robust and effective for different applications, through searching for retrieval functions that satisfy all the reasonable constraints. In the current work, we search only the neighborhood of existing retrieval functions. In the future, we plan to explore a general strategy to systematically search for an optimal retrieval function in the whole function space.

Query Adaptive Retrieval Models: Many studies in IR show that no single retrieval model is able to return satisfactory results for every query. The main reason is that the existing retrieval models fail to adjust their scoring functions dynamically based on queries. We plan to extend the current work on the axiomatic approach to design retrieval models that are able to automatically *adapt* to different queries based on their characteristics and inherent difficulty. Furthermore, the current search engines provide little support for a user to refine queries when the search results are poor. We plan to study interactive support to help users formulate better queries when necessary.

References

[1] M. Adriani. Using statistical term similarity for sense disambiguation in cross-language information retrieval. *Information Retrieval*, 2:69–80, 2000.

[2] J. Allan. Hard track overview in trec 2003 high accuracy retrieval from documents. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC2003)*, 2004.

[3] G. Amati and C. J. V. Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems*, 20(4):357–389, 2002.

[4] J. Bai, D. Song, P. Bruza, J.-Y. Nie, and G. Cao. Query expansion using term relationships in language models for information retrieval. In *Fourteenth International Conference on Information and Knowledge Management (CIKM 2005)*, 2005.

[5] A. Berger and J. Lafferty. Information retrieval as statistical translation. In *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–229, 1999.

[6] P. D. Bruza and T. Huibers. investigating aboutness axioms using information fields. In *Proceedings of the 1994 ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[7] G. Cao, J.-Y. Nie, and J. Bai. Integrating word relationships into language models. In *Proceedings of the 2005 ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[8] B. Carterette and J. Allan. Incremental test collections. In *Fourteenth International Conference on Information and Knowledge Management (CIKM 2005)*, 2005.

[9] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *Proceedings of the 2006 ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006.

[10] G. V. Cormack, C. R. Palmer, and C. L. Clarke. Efficient construction of large test collections. In *Proceedings of the 1998 ACM-SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[11] H. Fang, T. Tao, and C. Zhai. A formal study of information retrieval heuristics. In *Proceedings of the 2004 ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[12] H. Fang and C. Zhai. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the 2005 ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[13] N. Fuhr. Language models and uncertain inference in information retrieval. In *Proceedings of the Language Modeling and IR workshop.*

[14] N. Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[15] R. Fung and B. D. Favero. Applying Bayesian networks to information retrieval. *Communications of the ACM*, 38(3):42–48, 1995.

[16] J. Gao, J.-Y. Nie, H. He, W. Chen, and M. Zhou. Resolving query translation ambiguity using a decaying co-occurrence model and syntactic dependence relations. In *Proceedings of the 2002 ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002.

[17] J. Gao, J.-Y. Nie, E. Xun, J. Zhang, M. Zhou, and C. Huang. Improving query translation for cross-language information retrieval using statistical models. In *Proceedings of the 2001 ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[18] W. R. Grieff. A theory of term weighting based on exploratory data analysis. In *Proceedings of the 1998 ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[19] D. Harman and C. Buckley. Sigir 2004 workshop: Ria and where can ir go from here. *SIGIR Forum*, 38(2):45–49, 2004.

[20] F. Hartiwig and B. E. Dearing. *Exploratory Data Analysis.* Sage Publications, 1979.

[21] T. Huibers. Towards an axiomatic aboutness theory for information retrieval. *Information Retrieval, Uncertainty and Logics-Advanced Models for the representation and retrieval for information*, 1998.

[22] M.-G. Jang, S. H. Myaeng, and S. Y. Park. Using mutual information to resolve query translation ambiguities and query term weighting. In *Proceedings of the 37th annual meeting of the association for computational linguistics*, 1999.

[23] Y. Jing and W. B. Croft. An association thesaurus for information retreival. In *Proceedings of RIAO*, 1994.

[24] J. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems(NIPS)*, 2002.

[25] K. L. Kwok. A network approach to probabilistic information retrieval. *ACM Transactions on Office Information System*, 13:324–353, 1995.

[26] J. Lafferty and C. Zhai. Probabilistic relevance models based on document and query generation. In W. B. Croft and J. Lafferty, editors, *Language Modeling and Information Retrieval*. Kluwer Academic Publishers, 2003.

[27] V. Lavrenko and B. Croft. Relevance-based language models. In *Proceedings of SIGIR'01*, pages 120–127, Sept 2001.

[28] M. Lesk. Word-word associations in document retrieval systems. *American Documentation*, 20:27–38, 1969.

[29] D. Lin. An information-theoretic definition of similarity. In *Proceedings of International Conference on Machine Learning (ICML)*, 1998.

[30] S. Liu, F. Liu, C. Yu, and W. Meng. An effective approach to document retrieval via utilizing wordnet and recognizing phrases. In *Proceedings of the 2004 ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[31] P. Lyman and H. R. Varian. How much information? 2003, 2003. Retrieved from http://www.sims.berkeley.edu/research/projects/how-much-info-2003/ on Nov. 2005.

[32] A. Maeda, F. Sadat, M. Yoshikawa, and S. Uemura. Query term disambiguation for web cross-language information retrieval using a search engine. In *Proceedings of the fifth international workshop on information retrieval with Asian languages*, 2000.

[33] R. Mandala, T. Tokunaga, and H. Tanaka. Ad hoc retrieval experiments using wornet and automatically constructed theasuri. In *Proceedings of the seventh Text REtrieval Conference (TREC7)*, 1999.

[34] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM*, 7:216–244, 1960.

[35] M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proceedings of the 1998 ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[36] D. Moldovan and A. Novischi. Lexical chains for question answering. In *Proceedings of the 19th International Conference on Computational linguistics*, 2002.

[37] H. J. Peat and P. Willett. The limitations of term co-occurence data for query expansion in document retrieval systems. *Journal of the american society for information science*, 42(5):378–383, 1991.

[38] J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR'98*, pages 275–281, 1998.

[39] Y. Qiu and H. Frei. Concept based query expansion. In *Proceedings of the 1993 ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993.

[40] B. A. N. Ribeiro and R. Muntz. A belief network model for IR. In *Proceedings of SIGIR'96*, pages 253–260, 1996.

[41] B. Ribeiro-Neto, I. Silva, and R. Muntz. Bayesian network models for information retrieval. In F. Crestani and G. Pasi, editors, *Soft Computing in Information Retrieval: Techniques and Applications*, pages 259–291. Springer Verlag, 2000.

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

[43] S. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of SIGIR'94*, pages 232–241, 1994.

[44] S. Robertson and S. Walker. On relevance weights with little relevance information. In *Proceedings of SIGIR'97*, pages 16–24, 1997.

[45] S. E. Robertson. The probability ranking principle in ı∘. *Journal of Documentation*, 33(4):294–304, Dec. 1977.

[46] S. E. Robertson, S. Walker, S. Jones, M. M.Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. K. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*, pages 109–126, 1995.

[47] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall Inc., 1971.

[48] G. Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[49] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.

[50] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[51] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[52] G. Salton, C. S. Yang, and C. T. Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26(1):33–44, Jan-Feb 1975.

[53] M. Sanderson and H. Joho. Forming test collections with no system pooling. In *Proceedings of the 2004 ACM-SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[54] H. Schutze and J. O. Pedersen. A co-occurrence based thesaurus and two applications to information retrieval. *Information Processing and Management*, 33(3):307–318, 1997.

[55] S. Shi, J.-R. Wen, Q. Yu, R. Song, and W.-Y. Ma. Gravitation-based model for information retrieval. In *Proceedings of the 2005 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 488–495, 2005.

[56] A. Singhal. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–43, 2001.

[57] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the 1996 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, 1996.

[58] A. Singhal, J. Choi, D. Hindle, D. D. Lewis, and F. C. N. Pereira. ATT at TREC-7. In *Text REtrieval Conference*, pages 186–198, 1998.

[59] A. F. Smeaton and C. J. van Rijsbergen. The retrieval effects of query expansion on a feedback document retrieval system. *The Computer Journal*, 26(3):239–246, 1983.

[60] I. Soboroff, C. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgements. In *Proceedings of the 2001 ACM-SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[61] K. Sparck Jones. A statistical interpretation of term specifity and its application in retrieval. *Journal of Documentation*, 28(1):11–22, 1972.

[62] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments - part 1 and part 2. *Information Processing and Management*, 36(6):779–808 and 809–840, 2000.

[63] K. Sparck Jones and P. Willett, editors. *Readings in Information Retrieval.* Morgan Kaufmann Publishers, 1997.

[64] H. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

[65] C. J. van Rijbergen. A theoretical basis for theuse of co-occurrence data in information retrieval. *Journal of Documentation*, pages 106–119, 1977.

[66] C. J. Van Rijsbergen. *Information Retrieval.* Butterworths, 1979.

[67] C. J. van Rijsbergen. A non-classical logic for information retrieval. *The Computer Journal*, 29(6), 1986.

[68] E. Voorhees and D. Harman, editors. *Proceedings of Text REtrieval Conference (TREC1-9).* NIST Special Publications, 2001. http://trec.nist.gov/pubs.html.

[69] E. M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 1994 ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[70] E. M. Voorhees. Variations in relevance judgements and the measurement of retrieval effectiveness. In *Proceedings of the 1998 ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[71] E. M. Voorhees. Draft: Overview of the trec 2005 robust retrieval track. In *Notebook of the Thirteenth Text REtrieval Conference (TREC2005)*, 2005.

[72] E. M. Voorhees. Overview of the trec 2004 robust retrieval track. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC2004)*, 2005.

[73] K.-F. Wong, D. Song, P. Bruza, and C.-H. Cheng. Application of aboutness to functional benchmarking in information retrieval. *ACM Transactions on Information Systems*, 19(4):337–370, 2001.

[74] S. K. M. Wong and Y. Y. Yao. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):69–99, 1995.

[75] J. Xu and W. Croft. Query expansion using local and global document analysis. In *Proceedings of the SIGIR'96*, pages 4–11, 1996.

[76] C. Zhai and J. Lafferty. Model-based feedback in the KL-divergence retrieval model. In *Tenth International Conference on Information and Knowledge Management (CIKM 2001)*, pages 403–410, 2001.

[77] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR'01*, pages 334–342, Sept 2001.

[78] J. Zobel. How reliable are the results of large-scale information retrieval experiments? In *Proceedings of the 1994 ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[79] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*, 31(1):18–34, 1998.

Author's Biography

Hui Fang was born in Tianjin, China. She graduated from the Tsinghua University in 2001 with a B.S. degree in computer science. After that, she completed a M.S. degreee in computer science from the University of Illinois at Urbana-Champaign in 2004. Following the completion of her Ph.D., Hui Fang will begin work as an assistant professor in the Ohio State University.