© Copyright Yuanhua Lv 2012

IMPROVING THE EFFECTIVENESS OF LANGUAGE MODELING APPROACHES TO INFORMATION RETRIEVAL: BRIDGING THE THEORY-EFFECTIVENESS GAP

BY

YUANHUA LV

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, 2012

Urbana, Illinois

Doctoral Committee:

Associate Professor ChengXiang Zhai, Chair
Professor Jiawei Han
Assistant Professor Miles Efron
Dr. Evgeniy Gabrilovich, Google

To my wife Wan, my son William, and my daughter Adeline

Acknowledgments

First of all, I wish to express my deepest gratitude to my advisor, Dr. ChengXiang Zhai, for his support and guidance throughout my PhD study. It is Cheng who taught me the art of finding important research problems and the way of developing elegant and practical solutions. I could not make this far without him. I am also deeply grateful to him for his continuous encouragement and the maximum freedom he has given me in my research. I still remember during the early stage of my PhD study, Cheng encouraged and guided me to find the maximum overlap of my research interest and research strength to define this thesis topic; this has also significantly influenced both my research philosophy and my career choice. Cheng is definitely the one who has helped me grow from a newcomer into someone who has fallen in love with and will dedicate himself to this research area.

I would also like to extend my appreciation to the other members of my committee, Dr. Jiawei Han, Dr. Evgeniy Gabrilovich, and Dr. Miles Efron, for their efforts and constructive suggestions. Their comments are critical in making this thesis more complete and accurate. 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 the Text Information Management Group (TIMAN) and the Data and Information Systems Laboratory (DAIS), and have the opportunity to work with many colleagues from both groups. I enjoyed the time in discussing various problems with them, and the discussion definitely stimulated interesting ideas.

I would like to express my thanks to my mentors and colleagues at Yahoo! Labs and Microsoft Research. With them I had many valuable discussions of doing research to impact real world applications.

Finally, I would like to thank my dear wife Wan for her strong support through all these years to my study and career. Words can hardly express how fortunate I am for having her in my life. I am not only a happy husband but also a happy father: I would also like to thank our beloved son William and daughter Adeline, who have enlightened my life in so many ways. My acknowledgements cannot be finished without saying thanks to my parents and my parents-in-law for their care and love; in particular I am deeply grateful to my parents-in-law who have spent much time during the past two years helping take care of my son. This thesis is dedicated to them.

Abstract

The recent decade has witnessed an explosive growth of online information with the birth of Web. Search engines are by far the most powerful tools that help users find relevant information from large amounts of Web texts and have now become essential tools for all aspects of our life. The accuracy of a search engine is mainly determined by its retrieval model which formally specifies how each document matches a user's query. Improving the effectiveness of general retrieval models has been a long-standing difficult challenge in information retrieval research, yet is also a fundamentally important task, because an improved general retrieval model would benefit every search engine.

The language modeling approach to information retrieval has recently attracted much attention. In the language modeling approach, we assume that a query is a sample drawn from a language model: given a query Q and a document D, we compute the likelihood of "generating" query Q with a document language model estimated based on document D. We can then rank documents based on the likelihood of generating the query, i.e., query likelihood. On the one hand, with sound statistical foundation, the language modeling approach makes it easier to set and optimize retrieval parameters, and often outperforms traditional retrieval models. On the other hand, however, after more than one decade of research, the basic language modeling approach to retrieval still remains the same, mainly because the difficulty in accurately modeling the highly empirical notion of relevance within a standard statistical model has led to slow progress in optimizing language modeling approaches; this suggests that the theoretical framework of language models has a clear gap from what is needed to make a retrieval model empirically effective, a general problem we refer to as the "theory-effectiveness gap". We have identified the following theory-effectiveness gaps in current language modeling approaches:

First, one critical common component in any language modeling approach is the document language model. Traditional document language models follow the bag-of-words assumption that assumes term independence and ignores the positions of the query terms in a document. For example, in a query "computer virus", the occurrences of two query terms may be close to each other in one document (likely to mean computer virus) while far apart in another document (not necessarily about computer virus), which makes a huge difference for indicating relevance but is largely underexplored, suggesting the existence of a theory-effectiveness in standard document language models.

Second, accurate estimation of query language models plays a critical role in the language modeling approach

to information retrieval. Pseudo-relevance feedback (PRF) has proven very effective for improving query language models. The basic idea of PRF is to assume that a small number of top-ranked documents in the initial retrieval results are relevant and select from these documents useful terms to improve the query language model. However, existing PRF algorithms simply assume that all terms in a feedback document are equally useful, again ignoring term occurrence positions. They are often non-optimal, as a feedback document may cover multiple incoherent topics and thus contain many useless or even harmful terms. This shows a theory-effectiveness gap in estimating query language models based on PRF.

Third, although pseudo-relevance feedback approaches to the estimation of query language models can help improve the average retrieval precision, many experiments have shown that PRF often hurts many individual queries; the risk of PRF limits its usefulness in real search engines – another theory-effectiveness gap in query language models.

Fourth, the language modeling approach scores a document mainly based on the query likelihood score. A previously unknown deficiency of the query likelihood scoring function is that it is not properly lower-bounded for long documents. As a result of this deficiency, long documents which do match the query term can often be scored unfairly as having a lower relevancy than shorter documents that do not contain the query term at all. For example, for the aforementioned query "computer virus", a long document matching both "computer" and "virus" can easily be ranked lower than a short document matching only "computer". This reveals a clear theory-effectiveness gap between the standard query likelihood scoring function and the optimal way of scoring documents.

Fifth, the justification of using the basic query likelihood score for retrieval requires an unrealistic assumption, which states that the probability that a user who dislikes a document would use a query does not depend on the particular document. In reality, however, this assumption does not hold because a user who dislikes a document would more likely avoid using words in the document when posing a query. This theoretical gap between the basic query likelihood retrieval function and the notion of relevance suggests that the basic query likelihood function is a potentially non-optimal retrieval function.

To bridge the above theory-effectiveness gaps between the theoretical framework of standard language models and the empirical application of information retrieval, in this thesis, we clearly identified the causes of these gaps, and developed general methodologies to remove the causes from language models without destroying the statistical foundation and any other desirable properties of language models. Our explorations have delivered several more effective and robust general language modeling approaches, which can all be applied immediately to search engines to improve their ranking accuracy. Although this thesis focuses on language models, most of the proposed methodologies are actually more general, and can also be applied to retrieval models other than language models to bridge their theory-effectiveness gap as well.

Table of Contents

Chapter 1

Introduction

The recent decade has witnessed an explosive growth of online information with the birth of Web. The vast amount of data contains a lot of useful information for all kinds of human needs. Search engines, such as Google and Bing, are by far the most powerful tools that help users find relevant information from large amounts of Web texts and have now become essential tools for all aspects of our life. It is estimated that about 20 billion user queries from U.S. were submitted to search engines in November 2011 alone. Clearly, the effectiveness of search engines would significantly affect our productivity and quality of life.

Information Retrieval (IR) is, in brief, the underlying science of search engines. As a research field, IR research is primarily concerned with developing theories, principles, algorithms, and systems to help a user find relevant information from a collection of text documents to satisfy some information need of the user. IR research can be dated back to the 1950's [101]. In early days, the primary applications were library systems and the users were mostly librarians. However, the recent growth of online information, especially the development of the Web, has enabled ordinary people to be the users of various search engines.

Information 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 the notion of relevance, it is hard to find a clear boundary between relevant documents and non-relevant documents. Moreover, even if two documents are both 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 [84].

The key challenge of the information retrieval problem is to derive a retrieval model that can formally and appropriately specify how the content of each document matches a user's query, i.e., compute the relevance score of each document in response to a user's query. Although there are also other non-content or query independent features (e.g., links on the Web [9]) that can be exploited to improve relevance prediction, content-based matching remains the most important component in any search engine, and its performance can significantly affect the overall utility of a search engine. Thus, improving the effectiveness of retrieval models is a fundamentally important research problem

in information retrieval because an improved general retrieval model would benefit every search engine.

Improving the effectiveness of retrieval models has been a long-standing difficult challenges. Over the decades, many different retrieval models have been proposed, such as vector space models [89, 97], probabilistic inference models [107, 111], classical probabilistic retrieval models [85, 36, 86, 87], and language models [81, 121]. And the three most effective retrieval functions are the pivoted length normalization function from the vector space model [97], the BM25 function from the classic probabilistic model [87], and the query likelihood method with Dirichlet prior smoothing from the language model [121]. When optimized, these three models tend to all perform similarly well [29, 118]. However, we do not yet have a clear single winner among all the models that can consistently outperform all other models.

The language modeling approach to information retrieval has recently attracted much attention, due to its potential to develop an ultimately optimal retrieval model that is both theoretically sound and able to perform well empirically [118]. The language modeling approach to information retrieval was first introduced by Ponte and Croft in [81] and also independently explored in [77, 43]. In the language modeling approach, as shown in Figure 1.1, we assume that a query is a sample drawn from a language model: given a query Q and a document D, we compute the likelihood of "generating" query Q with a document language model estimated based on document D. We can then rank documents based on the likelihood of generating the query, i.e., query likelihood. With sound statistical foundation, the language modeling approach can leverage statistical estimation to optimize retrieval parameters, and often achieves comparable or better performance than a traditional model with less effort on parameter tuning. It can also be more easily adapted to model non-traditional and complex retrieval problems, including cross-lingual information retrieval [114, 60], distributed information retrieval [113, 95], structured document retrieval [79], personalized and context-sensitive search [94, 102], modeling redundancy [123], predicting query difficulty [24], expert finding [6, 33], passage retrieval [63], subtopic retrieval [119], etc.

However, on the other hand, the difficulty in accurately modeling the highly empirical notion of relevance within a standard statistical model has led to slow progress in optimizing language modeling approaches; after more than one decade of research, the basic language modeling approach to retrieval still remains the same [118]. This suggests that the theoretical framework of language models, without being able to accurately model the empirical notion of relevance, has a clear gap from what is needed to make a retrieval model empirically effective, a general problem we refer to as the "theory-effectiveness gap". This thesis reveals and identifies the following gaps in the current language modeling approaches:

- First, one critical common component in any language modeling approach to retrieval is the document language model, as shown in Figure 1.1. Traditional document language models follow the bag-of-words assumption that assumes term independence and ignores the positions of the query terms in a document. For example, in

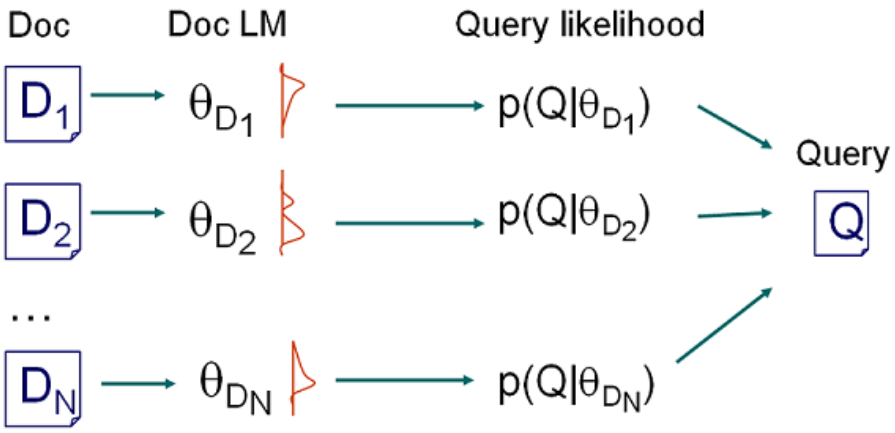

Figure 1.1: The query likelihood retrieval method

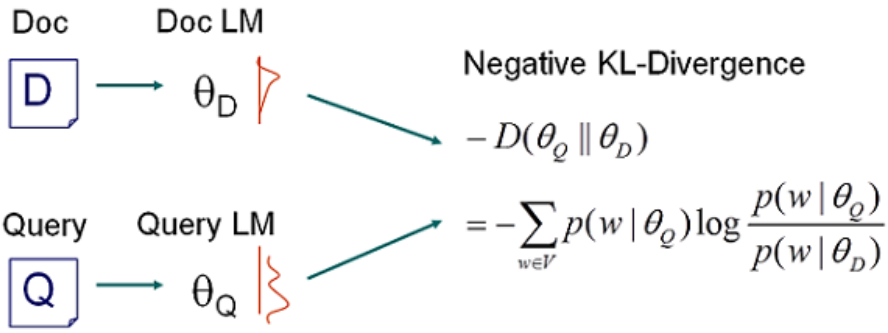

Figure 1.2: The KL-divergence retrieval method

a query "computer virus", the occurrences of two query terms may be close to each other in one document (likely to mean computer virus) while far apart in another document (not necessarily about computer virus), which makes a huge difference for indicating relevance but is largely underexplored, suggesting the existence of a theory-effectiveness in standard document language models.

- Second, in order to naturally and effectively support feedback, the KL-divergence retrieval model [121], which scores and ranks documents based on the negative KL-divergence between the query language model and the document language model, as shown in Figure1.2, has been proposed to generalize the query likelihood retrieval function so that feedback can be achieved through improved estimate of a query language model. Thus accurate estimation of query language models plays a critical role in the KL-divergence retrieval method. Pseudo-relevance feedback (PRF) has proven very effective for improving query language models [120, 61, 69]. The basic idea of PRF is to assume that a small number of top-ranked documents in the initial retrieval results are relevant and select from these documents useful terms to improve the query language model. However, existing

PRF algorithms simply assume that all terms in a feedback document are equally useful, again ingoring term occurrence positions. This is often non-optimal, as a feedback document may cover multiple incoherent topics and thus contain many useless or even harmful terms. This shows a theory-effectiveness gap in estimating query language models based on PRF.

- Third, although pseudo-relevance feedback approaches to the estimation of query language models can help improve the *average* retrieval precision, many experiments have shown that pseudo-relevance feedback often hurts many *individual* queries [22]; the risk of pseudo-relevance feedback limits its usefulness in real search engines – another theory-effectiveness gap in query language models.

- Fourth, the language modeling approach scores a document mainly based on the query likelihood score or the negative KL-divergence score. A previously unknown deficiency of the query likelihood scoring function and the KL-divergence scoring function is that they are not properly lower-bounded for long documents. As a result of this deficiency, long documents which do match the query term can often be scored unfairly as having a lower relevancy than shorter documents that do not contain the query term at all. For example, for the aforementioned query "computer virus", a long document matching both "computer" and "virus" can easily be ranked lower than a short document matching only "computer". This reveals a clear theory-effectiveness gap between the standard query likelihood scoring function and the optimal way of scoring documents.

- Fifth, the justification of using the basic query likelihood score for retrieval requires an unrealistic assumption, which states that the probability that a user who dislikes a document would use a query does not depend on the particular document [59]. In reality, however, this assumption does not hold because a user who dislikes a document would more likely avoid using words in the document when posing a query. This theoretical gap between the basic query likelihood retrieval function and the notion of relevance suggests that the basic query likelihood function is a potentially non-optimal retrieval function.

These problems of language models are mainly due to the fact that the standard statistical approach alone, without *directly* thinking about what makes a retrieval model practically effective, is only be able to model relevance inaccurately or inappropriately. To bridge the above heuristic or theoretical "gaps" between the theoretical framework of standard language models and the empirical application of information retrieval, in this thesis, we clearly identified the causes of these gaps, and developed general methodologies to remove the causes from language models without destroying their statistical foundation to improve language models from different perspectives, corresponding to the three main components of language modeling approaches as shown in Figure 1.2, i.e., document language models, query language models, and the query likelihood scoring function (or the KL-divergence scoring function):

- **Positional document language models:** existing studies have not really incorporated term position and proximity evidence into language models as an inner component. To break this limitation, we propose a novel positional language model (PLM) which implements both heuristics in a unified language model to bridge this theory-effectiveness gap. The key idea is to define a language model for each position of a document, and score a document based on the scores of its positional language models. The PLM is estimated based on propagated counts of words within a document through a proximity-based kernel function. Specifically, we let each word at each position of a document propagate the evidence of its occurrence to all other positions in the document so that positions close to the word would get more share of the evidence than those far away, which both captures the proximity heuristic and achieves an effect of "soft" passage retrieval. In addition, we work out a mathematical approach to reduce the computational complexity dramatically. The PLM has shown more robust and effective than the general document language model, the general passage retrieval, and a state-of-the-art proximity retrieval model.

- **Positional relevance models for estimating query language models**: the existence of multiple topics and irrelevant information would lead potentially harmful terms from non-relevant topics (e.g., the ad text) to be picked up as feedback terms in pseudo-relevance feedback methods for estimating query language models. Thus a critical challenge in improving all pseudo-relevance feedback methods is to effectively select from feedback documents those terms that are most likely relevant to the query topic. We tackle this challenge by exploiting the position and proximity information of terms as cues to assess if a term is related to the query topic. Since topically related content is usually grouped together in documents, terms closer to the occurrences of query words are, in general, more likely relevant to the query topic, thus a good feedback model should intuitively place higher weights on such terms. Based on this intuition, we propose a novel positional relevance model to incorporate the cues of term positions and term proximity in a probabilistic approach based on the proposed positional language model for pseudo-relevance feedback. An important advantage of the this method is that it can model the "relevant positions" in a feedback document with probabilistic models so as to assign more weights to terms at more relevant positions in a principled way, thus leading naturally to selection of expansion terms more likely relevant to the query topic. Besides, we develop two methods to estimate the proposed positional relevance model based on different sampling processes. Experiment results show that the proposed method is more effective and robust than current state-of-the-art pseudo-relevance feedback approaches.

- **Direct optimizing the robustness of pseudo-relevance feedback approaches to estimating query language models**: pseudo-relevance feedback is risky due to its pseudo nature [22]. An important, yet difficult problem is to optimize the overall effectiveness of pseudo-relevance feedback without sacrificing the performance of individual queries too much. To address this problem, we propose a novel learning algorithm based on the boosting

framework to optimize pseudo-relevance feedback through combining a set of basis (weak) feedback algorithms using a loss function defined to directly measure both robustness and effectiveness. Specifically, like all other boosting algorithms, our algorithm iteratively selects and combines basis feedback methods. In each iteration, a basis feedback method is selected to improve those queries on which the already selected basis feedback methods perform poorly in terms of both effectiveness and robustness. At last, we use a linear combination of these basis feedback methods as its final feedback model. The proposed algorithm can potentially accommodate many basis feedback methods, including the proposed pseudo-relevance feedback methods in this thesis, as features in the model, making the proposed method a general optimization framework for pseudo-relevance feedback. The experiment results demonstrate that our algorithm can achieve better average precision and meanwhile dramatically reduce the number and magnitude of feedback failure cases. Moreover, the proposed algorithm is actually more general and applicable to pseudo-relevance feedback in other retrieval models as well.

- **Lower-bounding the query likelihood scoring function**: one key component in the query likelihood function (as well as in the KL-divergence function and other state-of-the-art retrieval functions) is term frequency normalization by document length [121]. This component captures the following heuristic: a document should be scored higher if it contains more occurrences of a query term, but the term frequency signal, if applied alone, would have a tendency to overly reward long documents due to their high likelihood of matching a query term more times than a short document, so term frequency should be regularized by document length. Our analysis has shown that the improper lower-bound of the query likelihood scoring function is caused by its improper lower-bound of term frequency normalization. In order to analytically diagnose this problem, we propose two desirable formal constraints to capture the heuristic of lower-bounding term frequency normalization, and use constraint analysis to examine the query likelihood function. Analysis results show that it can only satisfy the constraints for a certain range of parameter values and/or for a particular set of query terms. Empirical results further show that the retrieval performance tends to be poor when the parameter is out of the range or the query term is not in the particular set. To solve this problem, we propose an efficient method to introduce a sufficiently large lower bound for term frequency normalization which can be shown analytically to fix the problem. Our experimental results demonstrate that the proposed method, incurring almost no additional computational cost, can improve the retrieval performance of the query likelihood function significantly. In fact, our empirical observation and constraint analysis show that the problem of improper lower-bound of term frequency normalization is a common deficiency of current retrieval models. And the proposed method is a general solution that can be used as a plug-and-play patch to multiple state-of-the-art retrieval models to improve their effectiveness.

- **Query likelihood with negative query generation:** the basic query likelihood function is a potentially non-optimal retrieval function, because it makes an unrealistic assumption to ignore the probability of generating

a "negative query" from a document. We attempt to improve the query likelihood function by bringing back the negative query generation. Specifically, we exploit document D to infer the "negative queries" that a user would use to avoid retrieving D based on the intuition that such queries would not likely have any information overlap with D. We then propose an effective and efficient approach to estimate probabilities of negative query generation based on the principle of maximum entropy, and derive a more complete query likelihood retrieval function with the negative query generation component, which essentially scores a document with respect to a query according to the ratio of the probability that a user who likes the document would pose the query to the probability that a user who dislikes the document would pose the query. In addition, we further develop a more general probabilistic distance retrieval method to naturally incorporate query language models, which covers the proposed query likelihood with negative query generation as its special case. The proposed approach not only bridges the theoretical gap between the standard query likelihood and the probability ranking principle, but also improves retrieval effectiveness over the standard query likelihood with no additional computational cost. More interestingly, the developed query likelihood with negative query generation leads to the same ranking formula as derived by lower-bounding the query likelihood scoring function, thus essentially providing a probabilistic interpretation for the heuristic method of lower-bounding term frequency normalization in the basic query likelihood method.

Developing an optimal retrieval function has huge impact, because it will improve the accuracy of every search engine. This thesis proposes to improve the effectiveness language modeling approaches to information retrieval through bridging the "theory-effectiveness gap", and has resulted in several more effective and robust retrieval algorithms that are as efficient as standard language modeling approaches. Although we focus on language models in this thesis, most of the proposed methodologies are actually not restricted to language models and can also be applied to retrieval models other language models. All the proposed new models are general, and thus can be used immediately in any search engine to improve its retrieval accuracy over the current retrieval models.

The rest of the thesis is organized as follows. First, we discuss the literature of language modeling approaches to information retrieval in Chapter 2. Then, we present positional document language models in Chapter 3. Next, we introduce a novel positional relevance model for estimating query language models in Chapter 4. We further propose a novel boosting approach, and discuss how we can apply it to improve not only the effectiveness but also the robustness of pseudo-relevance feedback approaches for estimating query language models in Chapter 5. After that, in Chapter 6 and Chapter 7, we present two extended query likelihood retrieval functions, through lower-bounding term frequency normalization and bringing back the negative query generation, respectively. Finally, we summarize the contributions of the thesis and discuss future work in Chapter 8.

Chapter 2

Language Modeling Approaches to Information Retrieval

2.1 Statistical Language Modeling

A statistical language model, or simply a language model, is a probability distribution over word sequences. The first serious statistical language modeler was contributed by Shannon [93], where he used n-grams to investigate the information content of English text [64]. This seminal work opened up a whole research branch of language modeling, as it provides a principled way to quantify the uncertainties associated with the use of natural language. Since then, statistical language modeling has become one of the main techniques used in the speech recognition tasks for many years, and it also gradually expended itself successfully into speech recognition [46], machine translation [10], information retrieval [81], and other research areas.

Given a language model, we can sample word sequences according to the distribution to obtain a text sample. In this sense, we may use a language model to "generate" text. Thus, a language model is also often regarded a probabilistic mechanism for generating text, i.e., a generative model for text. In text applications, a statistical language model is usually constructed on words with necessary stemming. We name such units terms. If we enumerate all the possible sequences of words and give a probability to each sequence, the model would be too complex to estimate because the number of parameters is potentially infinite since we have potentially infinite number of word sequences. That is, we would never have enough data to estimate these parameters. Thus, we always have to make assumptions to simplify the model. The simplest language model is the unigram language model in which we assume that a word sequence is generated by generating each word independently. Thus, the probability of a sequence of words would be equal to the product of the probability of each word. Formally, let V be the set of words in the vocabulary, and $w_1 \cdots w_n$ a word sequence, where $w_i \in V$ is a word. We have:

$$p(w_1 \cdots w_n) = \prod_{i=1}^{n} p(w_i) \tag{2.1}$$

It is easy to see that given a unigram language model θ, we have as many parameters as the words in the vocabulary, i.e., $\{p(w_i|\theta)\}_{i=1}^{|V|}$.

Now suppose we have observed a document D, which is assumed to be generated using a unigram language model θ, and we would like to infer θ (i.e., estimate the probability of each word w, $p(w|\theta)$) based on the observed D. This is a standard problem in statistics and can be solved using many different methods. One popular method is the maximum likelihood (ML) estimator, which seeks a model $\hat{\theta}$ that would give the observed data the highest likelihood:

$$\hat{\theta} = \arg\max_{\theta} p(D|\theta) \tag{2.2}$$

Unigram language models clearly make unrealistic assumptions about word occurrence dependence in text, since they assume that each word is generated independently. More sophisticated language models have thus been developed to address the limitations of unigram language models. For example, an n-gram language model would capture some limited dependency between words and assume the occurrence of a word depends on the proceeding $n-1$ words. As a specific example, a bigram language model is defined as follows:

$$p(w_1 \cdots w_n) = p(w_1) \prod_{i=2}^{n} p(w_i|w_{i-1}) \tag{2.3}$$

Such a bigram language model can capture any potential local dependency between two adjacent words.

While theoretically speaking, we would like to adopt a sophisticated language model that can model our language more accurately, in reality, we often have to make a tradeoff. This is because as the complexity of a language model increases, so does the number of parameters. As a result, we would need much more data to estimate the parameters. With limited amount of data, our estimate of parameters would not be accurate. The computational cost of complex language models is also a concern for all large-scale retrieval applications. So far, the simplest unigram language model has been shown to be quite effective for information retrieval, while more sophisticated language models such as bigram language models or trigram language models tend not to improve much over the unigram language model. Throughout the whole thesis, our discussion is on the basis of unigram models unless otherwise stated.

In the following sections, we summarize representative language modeling approaches to information retrieval.

2.2 The Query Likelihood Retrieval Method

The query likelihood retrieval method was first introduced by Ponte and Croft in [81]. In this method, given a query Q and a document D, we compute the likelihood of "generating" query Q with a model θ_D estimated based on document D, and then score and rank the document based on the likelihood of generating the query:

$$Score(D, Q) = p(Q|\theta_D) \tag{2.4}$$

The query generation can be based on any language model. Different models make different assumptions about term occurrences. So far, using a multinomial distribution [77, 43, 121] for θ_D has been most popular and most successful, which is also adopted in this thesis. However, several other choices have also been explored, including the multiple Bernoulli distribution [81, 76], the multiple Poisson distribution [72], and the hypergeometric distribution [106]. With the multinomial distribution, the query likelihood is

$$p(Q|\theta_D) = \prod_w p(w|\theta_D)^{c(w,Q)} \tag{2.5}$$

where $c(w, Q)$ is the count of term w in query Q.

According to the maximum likelihood estimator, we have the following estimation of the document language model θ_D for the multinomial model:

$$p_{ml}(w|\theta_D) = \frac{c(w, D)}{|D|} \tag{2.6}$$

where $c(w, D)$ represents the count of term w in document D, and $|D|$ is the document length. The document language model θ_D needs to be smoothed to overcome the zero-probability problem, and an effective method is the Dirichlet prior smoothing [121]:

$$p(w|\theta_D) = \frac{|D|}{|D| + \mu} p_{ml}(w|D) + \frac{\mu}{|D| + \mu} p(w|C) \tag{2.7}$$

Here $p(w|C)$ is the collection language model and is estimated as $p(w|C) = \frac{c(w,C)}{\sum_{w'} c(w',C)}$, where $c(w, C)$ indicates the count of term w in the whole collection C, and μ is a smoothing parameter (Dirichlet prior) which is usually set empirically. Smoothing plays two different roles in the query likelihood retrieval method [121]: one role is to assign non-zero probabilities to terms that are not observed in the document, and the other role is to weaken the effect of non-discriminative terms in the query to achieve an "IDF" effect.

Assuming the Dirichlet prior smoothing method, we can rewrite the query likelihood scoring function as follows [42, 121]:

$$\log p(Q|\theta_D) \overset{rank}{=} \sum_{w \in Q \cap D} c(w, Q) \log \left(1 + \frac{c(w, D)}{\mu p(w|C)}\right) + |Q| \log \frac{\mu}{|D| + \mu} \tag{2.8}$$

where $|Q|$ represents query length. It shows that, although the query likelihood method is motivated in a different way than a traditional model such as the vector-space model, it tends to boil down to retrieval functions that implement retrieval heuristics (such as TF-IDF weighting and document length normalization) similar to those implemented in a

traditional model [42, 121].

In the past decade, many more complex variants of the query likelihood method have been proposed for ad hoc retrieval. For example, n-gram [98] and dependence language model [37] have been explored to go beyond the bag-of-word assumption; the query likelihood was also extended as a translation model to allow inexact matching of semantically related words [8]; a full Bayesian query likelihood was studied to consider uncertainty of an estimation of θ_D [117]; parsimonious language models was proposed to improve the discrimination of language models [44]; cluster-based smoothing methods were evaluated for document-specific smoothing [65, 110, 103], etc. Although these extensions often outperform the basic query likelihood, they tend to incur significantly more computational cost.

The query likelihood method has also been shown to perform well for a variety of retrieval tasks, including ad-hoc retrieval [81, 121, 58], cross-lingual information retrieval [114, 60], distributed information retrieval [113, 95], structured document retrieval [79], personalized and context-sensitive search [94, 102], modeling redundancy [123], predicting query difficulty [24], expert finding [6, 33], passage retrieval [63], subtopic retrieval [119], etc.

2.3 The Notion of Relevance in Language Modeling Approaches

To better understand the retrieval foundation of the query likelihood method, Lafferty and Zhai [59] provided a general relevance-based derivation of the query likelihood method. Formally, let random variables D and Q denote a document and query, respectively. Let R be a binary random variable that indicates whether D is relevant to Q or not. Following Sparck Jones et al. [50], we will denote by ℓ ("like") and $\bar{\ell}$ ("not like") the value of the relevance variable. The Probability Ranking Principle [84] provides a justification for ranking documents for a query based on the conditional probability of relevance, i.e., $p(R = \ell|D, Q)$. This is equivalent to ranking documents based on the odds ratio, which can be further transformed using Bayes' Rule:

$$O(R = \ell|Q, D) = \frac{p(R = \ell|Q, D)}{p(R = \bar{\ell}|Q, D)} \propto \frac{p(Q, D|R = \ell)}{p(Q, D|R = \bar{\ell})} \tag{2.9}$$

There are two different ways to decompose the joint probability $p(Q, D|R)$, corresponding to "document generation" and "query generation" respectively. With document generation $p(Q, D|R) = p(D|Q, R)p(Q|R)$, we have

$$O(R = \ell|Q, D) \propto \frac{p(D|Q, R = \ell)}{p(D|Q, R = \bar{\ell})} \tag{2.10}$$

Most classical probabilistic retrieval models [85, 50, 36] are based on document generation. Fuhr [36] has provided in-depth discussions in this direction.

Query generation, $p(Q, D|R) = p(Q|D, R)p(D|R)$, is the focus of this paper. With query generation, we end up

with the following ranking formula:

$$O(R = \ell|Q, D) \propto \frac{p(Q|D, R = \ell)p(R = \ell|D)}{p(Q|D, R = \bar{\ell})p(R = \bar{\ell}|D)} \tag{2.11}$$

in which, the term $p(R|D)$ can be interpreted as a prior of relevance on a document, which can be used to encode any bias on documents. Without such extra knowledge, we may assume that this term is the same across all the documents and obtain the following simplified ranking formula:

$$O(R = \ell|Q, D) \propto \frac{p(Q|D, R = \ell)}{p(Q|D, R = \bar{\ell})} \tag{2.12}$$

There are two components in this model. $p(Q|D, R = \ell)$ can be interpreted as a positive query generation model. It is essentially the basic query likelihood, which suggests that the query generation probability used in all the query likelihood scoring methods intuitively means the probability that a user who likes document D would pose query Q. Another component $p(Q|D, R = \bar{\ell})$ can be interpreted as the generation probability of a "negative query" from a document, i.e., the probability that a user who dislikes a document D would use a query Q. In order to justify using the basic query likelihood alone as the ranking formula, an assumption has to be made about this negative query generation component, which states that the probability of negative query generation does not depend on the particular document [59], formally

$$p(Q|D, R = \bar{\ell}) = p(Q|R = \bar{\ell}) \tag{2.13}$$

This assumption enables ignoring the negative query generation in the derivation of the basic query likelihood retrieval function, leading to the following basic query likelihood scoring method: $O(R = \ell|Q, D) \propto p(Q|D, R = \ell) = P(Q|\theta_D)$.

2.4 The KL-Divergence Retrieval Method

A major deficiency of the query likelihood method is that it cannot easily incorporate relevance or pseudo-relevance feedback [120]. To address this problem, a probabilistic distance model called Kullback-Leibler (KL) divergence retrieval method was proposed [58]. The KL-divergence method can actually cover the query likelihood retrieval model as a special case when the query model is estimated based on only the query. Moreover, the development of the KL-divergence retrieval model [58], which explicitly models both document and query language models, has attracted many efforts to propose effective pseudo-relevance feedback methods for improving the estimate of query language

models, e.g., [58, 61, 120, 57, 28, 103, 23, 69].

In the KL-divergence retrieval model [58], queries and documents are all represented by unigram language models. Assuming that these language models can be estimated appropriately, the KL-divergence retrieval model scores a document D with respect to a query Q by computing the negative Kullback-Leibler divergence between the query language model θ_Q and the document language model θ_D:

$$S(Q,D) = -D(\theta_Q||\theta_D) = -\sum_{w \in V} P(w|\theta_Q) \log \frac{P(w|\theta_Q)}{P(w|\theta_D)} \tag{2.14}$$

where V is the set of words in our vocabulary. Clearly, the retrieval performance of the KL-divergence model would depend on how we estimate the document language model θ_D and the query language model θ_Q. The document language model θ_D can be estimated using Formula 2.7. The query language model θ_Q intuitively captures what the user is interested in, and thus would affect retrieval accuracy significantly. Without feedback, query language models are often estimated by using the MLE method on the query text:

$$P(w|\theta_Q) = \frac{c(w,Q)}{|Q|} \tag{2.15}$$

where $c(w,Q)$ is the count of word w in query Q, and $|Q|$ is the total number of words in the query.

2.5 Query Language Models with Pseudo-Relevance Feedback

The most effective methods for estimating query language models generally rely on the strategy of pseudo-relevance feedback (PRF) [88, 85, 91, 11, 87, 61, 120], which can improve retrieval performance significantly over simple estimation methods that only use the query [69]. The basic idea of estimating query language models with PRF is to assume that a small number of top-ranked documents $F = \{D_1 \dots D_{|F|}\}$ in the initial retrieval results are relevant and select from these documents useful terms to re-estimate a more accurate query language model θ'_Q. Several popular methods, e.g., the relevance models [61, 1] and the simple two-component mixture model [120], have been shown to be robust and effective to improve the estimation of the query model in the setting of pseudo-relevance feedback (i.e., estimating θ'_Q with F). We now review these representative methods for estimating query language models.

2.5.1 Relevance Models

The relevance model (RM) was developed by [61]. RM does not explicitly model feedback. Instead, it models a more generalized notion of relevance. Given a query Q, a relevance model is a multinomial distribution $P(w|Q)$ that encodes the likelihood of each term w given the query as evidence. We now describe two methods proposed in [61]

for estimating the relevance model.

In the first method (often called RM1), the authors first compute the joint probability of observing a word together with the query words in each feedback document and then aggregate the evidence by summing over all the documents. It essentially uses the query likelihood $P(Q|D)$ as the weight for document D and takes an average of the probability of word w given by each document language model. Formally, let Θ represent the set of smoothed document models in the pseudo feedback collection F and $Q = \{q_1, q_2, \cdots, q_m\}$. The formula of RM1 is derived as follows:

$$\begin{aligned} P_{rm1}(w|Q) &= \sum_{\theta_D \in \Theta} P(w|\theta_D) P(\theta_D|Q) \\ &= \sum_{\theta_D \in \Theta} P(w|\theta_D) \frac{P(Q|\theta_D)P(\theta_D)}{P(Q)} \\ &\propto \sum_{\theta_D \in \Theta} P(w|\theta_D) P(Q|\theta_D) P(\theta_D) \\ &= \sum_{\theta_D \in \Theta} P(w|\theta_D) P(\theta_D) \prod_{i=1}^{m} P(q_i|\theta_D) \end{aligned} \tag{2.16}$$

In the derivation, for the posterior of document language model $P(\theta_D|Q)$, we can rewrite it as being proportional to $P(\theta_D) \prod_{i=1}^{m} P(q_i|\theta_D)$ by the Bayes' rule, in which the second term $\prod_{i=1}^{m} P(q_i|\theta_D)$ is precisely the query likelihood given the document, which tells us how likely the document is relevant to the query, while the first term $p(\theta_D)$ is a general prior on documents and is often assumed to be uniform without any additional prior knowledge about document D. Thus, the count of a word in a highly scored document according to the query likelihood retrieval model would be weighted more when combining the counts, which intuitively makes sense.

The Dirichlet prior smoothing method is used to smooth the language model of each pseudo-relevant document in F for both RM1 and RM2 (which we will introduce later):

$$P(w|\theta_D) = \frac{c(w, D) + \mu_{fb} \cdot P(w|\mathcal{C})}{|D| + \mu_{fb}} \tag{2.17}$$

where μ_{fb} is another Dirichlet prior different from the one used in the initial retrieval step (Formula 2.7). However, in some popular information retrieval systems, e.g., Lemur toolkit and Indri search engine [1], to simplify the implementation of RM1, the query likelihood score $\prod_{i=1}^{m} P(q_i|\theta_D)$ in Formula 2.16 is taken or transformed directly from the initial retrieval results; while only $P(w|\theta_D)$ in Formula 2.16 is smoothed using Formula 2.17. In practice, such an estimation shows a slightly better performance. Thus this strategy is also adopted in our study.

In the second method (i.e., RM2), the authors compute the association between each word and the query using documents containing both query terms and the word as "bridges". The strongly associated words are then assigned

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

high probabilities in the relevance model. Formally, the derivation is as follows:

$$\begin{aligned} P_{rm2}(w|Q) &= \frac{P(Q|w)P(w)}{P(Q)} \\ &\propto P(Q|w)P(w) \\ &= P(w)\prod_{i=1}^{m} P(q_i|w) \\ &= P(w)\prod_{i=1}^{m}\sum_{\theta_D\in\Theta} P(q_i|\theta_D)P(\theta_D|w) \\ &= P(w)\prod_{i=1}^{m}\sum_{\theta_D\in\Theta} P(q_i|\theta_D)\frac{P(w|\theta_D)P(\theta_D)}{P(w)} \end{aligned} \tag{2.18}$$

where $p(w)$ can be estimated as:

$$P(w) = \sum_{\theta_D\in\Theta} P(w|\theta_D)P(\theta_D) \tag{2.19}$$

where in Formula 2.18 and 2.19, $P(\theta_D)$ is also kept uniform and θ_D is smoothed using Formula 2.17.

Indeed, we see that the two estimation methods RM1 and RM2 mainly differ in how they aggregate the evidence of a word w co-occurring with query words: RM1 first aggregates the evidence for all the query words by taking a product, and then further aggregates the evidence by summing over all the possible document models, while RM2 does the opposite.

The relevance model $P(w|Q)$ can be interpolated with the original query model θ_Q to improve performance [1]. In this paper, we will only evaluate the following two interpolated versions of the relevance model, called RM3 and RM4 respectively:

$$\text{RM3:}\quad P(w|\theta'_Q) = (1-\alpha)\cdot P(w|\theta_Q) + \alpha\cdot P_{rm1}(w|Q) \tag{2.20}$$

$$\text{RM4:}\quad P(w|\theta'_Q) = (1-\alpha)\cdot P(w|\theta_Q) + \alpha\cdot P_{rm2}(w|Q) \tag{2.21}$$

where $\alpha \in [0,1]$ is a parameter to control the amount of feedback information. Such an interpolation coefficient α is also employed explicitly or implicitly in other pseudo-relevance feedback methods as will be introduced below.

2.5.2 Simple Mixture Model

The simple mixture feedback model (SMM) proposed by [120] fits the feedback documents F with a two-component mixture model, where one component is a fixed background language model $P(w|C)$ estimated using the collection and the other is an unknown, to-be-discovered topic model $P(w|\theta_F)$. Essentially, the words in F are assumed to be drawn from two models:(1) background model $P(w|C)$ and (2) topic model $P(w|\theta_F)$. Specifically, when we generate a word using this mixture model, we would first decide which model to use and then sample a word using the chosen model. Thus, the probability of generating a word w is:

$$P(w) = (1-\lambda) \cdot P(w|\theta_F) + \lambda \cdot P(w|C) \tag{2.22}$$

where $\lambda \in [0,1]$ is the probability of choosing the background model $P(\cdot\,|C)$ to generate the word. Thus the log-likelihood function for the entire set of feedback documents is:

$$\log P(F|\theta_F) = \sum_{w \in V} c(w,F) \log((1-\lambda) \cdot P(w|\theta_F) + \lambda \cdot P(w|C)) \tag{2.23}$$

where $c(w,F)$ is the count of word w in the set of feedback documents. Since $c(w,F) = \sum_{D \in F} |D| \cdot P(w|D)$, it means that the document length $|D|$ is used as a weight for document D to sum over all evidence from each feedback document. As a result, long documents are favored.

Intuitively, λ indicates how much weight we want to put on the background model. Note that in attempting to find the optimal θ_F using maximum likelihood, we need to set λ to a fixed value [120]. In effect, $P(w|C)$ together with λ would "encourage" the estimated topic model θ_F to focus more on the words with small probabilities in $P(w|C)$. That is, the SMM method tends to assign high probabilities to words with high "IDF" scores.

The estimate of θ_F can be computed using the Expectation-Maximization (EM) algorithm [27]. Specifically, in the E-step, we would use the following equation to compute the posterior probability of a word w being generated using θ_F based on the current estimation of θ_F:

E-Step:

$$P(z_w = 1) = \frac{(1-\lambda) \cdot P^{(n)}(w|\theta_F)}{(1-\lambda) \cdot P^{(n)}(w|\theta_F) + \lambda \cdot P(w|C)} \tag{2.24}$$

where $z_w \in \{0,1\}$ is a hidden variable indicating whether word w is generated using the topic model θ_F (i.e., $z_w = 1$) or the collection model $P(w|C)$ (i.e., $z_w = 0$).

Intuitively, if $P(w|\theta_F)$ is much larger than $P(w|C)$, we would "guess" that w is more likely generated using θ_F, and $P(z_w = 1)$ would be high. Then, in the M-step, we use the following equation to update the estimate of θ_F:

M-Step:

$$P^{(n+1)}(w|\theta_F) = \frac{c(w,F)P(z_w = 1)}{\sum_{w' \in V} c(w',F)P(z_{w'} = 1)} \tag{2.25}$$

Similarly, the query language model is updated by interpolating θ_F with the original query model θ_Q using a coefficient α.

2.6 Summary

In this chapter, we summarized the language modeling approaches to information retrieval. As compared to traditional retrieval models, the language modeling approach has clear statistical foundation, which makes it easier to set and optimize retrieval parameters based on standard statistical estimation. However, the difficulty in accurately modeling the highly empirical notion of relevance within a standard statistical model has also led to slow progress in optimizing language modeling approaches; after more than one decade of research, the basic language modeling approach to retrieval still remains unchanged: the query likelihood retrieval method or the KL-divergence retrieval method as the retrieval function, Dirichlet prior smoothing for the estimation of document language models, and the relevance model or the simple mixture model for the estimation of query language models.

This thesis aims at bridging this "theory-effectiveness gap" between the theoretical framework of standard language models and the empirical application of information retrieval, so as to allow the language modeling approach to more directly and accurately model what is needed to make a retrieval model empirically effective. Specifically, we clearly identified the causes of existing theory-effectiveness gaps, and developed general methodologies to remove the causes from language models without destroying the statistical foundation and any other desirable properties of language models.

Chapter 3

Positional Language Models

3.1 Introduction

In Section 2.2, the analysis of the language modeling retrieval formula shows that although language models are motivated in a different way than a traditional model such as the vector-space model, they tend to boil down to retrieval functions that implement retrieval heuristics similar to those implemented in a traditional model, such as TF-IDF weighting and document length normalization [121]. One important advantage of the language modeling approach is that it can leverage statistical estimation to optimize retrieval parameters [118].

However, on the other hand, the standard statistical language modeling framework makes it hard to incorporate additional retrieval heuristics without destroying the statistical foundation of language modeling approaches. Two particularly important retrieval heuristics that are still remaining "external" to the language modeling approach are the term proximity heuristic and the passage retrieval heuristic: (1) proximity heuristic which rewards a document where the matched query terms occur close to each other; (2) passage retrieval which scores a document mainly based on the best matching passage.

Although much work has been done in language models, traditional document language models follow the bag-of-words assumption that assumes term independence and ignores the positions of the query terms and the passage evidence in a document. Or at the best, some existing studies have only attempted to use a standard language model as a black box to implement these heuristics, which means that these heuristics have not really been incorporated into a language model as a component and make it hard to leverage advantages of language models to optimize the combination parameters. For example, proximity heuristic has been studied in [104] where the authors proposed heuristic proximity measures and combined them with the scores of documents computed using standard language models. Also, passage retrieval has been studied in [63] where the authors explored different ways of segmenting text to create passages and then applied standard language models on top of the passages as if they were regular documents. A common deficiency of these studies is that the proximity and passage retrieval heuristics are not modeled from language modeling perspective, which makes it difficult to optimize the way of combining them with language models, suggesting the existence of a clear "gap" between the existing language modeling approaches and what is

needed to make a retrieval model empirically effective.

In this chapter, we propose a novel positional language model (PLM) which implements both heuristics in a unified language model. The key idea is to define a language model for each *position* of a document (thus the name positional language model), and score a document based on the scores of its PLMs. This is in contrast with virtually all the existing work in which a document language model is generally defined for the *entire* document. An important advantage of introducing a language model for each position is that it can allow us to model the "best-matching position" in a document with probabilistic models, thus supporting "soft" passage retrieval naturally.

It is nontrivial to estimate PLM at a position of a document, because each position only contains a single term: PLM will suffer from a serious data sparseness problem if we estimate it purely based on the position content. To address this problem, in our work, the PLM would be estimated using a novel smoothing strategy based on the propagated word counts from the words at all other positions in the document. Specifically, we let each word at each position of a document to propagate the evidence of its occurrence to all other positions in the document so that positions close to the word would get more share of the evidence than those far away. This way, each position would receive propagated counts of words from all the words in the document with most propagated counts coming from words near the position. We can then estimate a language model for the position based on the propagated counts reaching the position.

A main technical challenge in implementing this idea is how to define the propagation function and estimate the PLM accordingly. We propose and evaluate several different proximity-based kernel functions for propagation. With some specific choices, we show that the PLM can cover the standard whole document language model and the fixed-window passage language model, as special cases. Since in all these kernel functions, close-by positions would receive more propagated counts than positions far away from the current word, the PLM also captures the proximity heuristics.

Once we have a language model estimated for each position, we can use one or multiple PLMs of a document as regular document language models to generate a score for the document. We propose and study three general document ranking strategies for combining different PLMs to score documents, including scoring based on the best PLM, combining scores from PLMs at multiple positions, and combining PLMs with different kernel ranges.

Experiment results on several standard test collections show that among all the proximity-based kernel functions, the Gaussian density kernel performs the best, and that combining PLMs with different propagation ranges is the best document ranking strategy. It is also observed that the proposed PLM not only outperforms the general document language model, but also outperforms the regular sliding-window passage retrieval method and a state-of-the-art proximity-based retrieval model. Overall, the PLM is shown to be able to achieve "soft" passage retrieval and capture proximity heuristic effectively in a unified probabilistic framework.

3.2 Related Work

Term proximity in information retrieval has been previously studied in [54, 55, 40, 20, 82, 74, 14, 15, 104, 99]. Keen's work [54, 55] is among the earliest efforts, in which, a "NEAR" operator was introduced to address proximity in Boolean retrieval model. The shortest interval containing a match set was first used as a measure of proximity in [20, 40]. Recent work has attempted to heuristically incorporate proximity into an existing retrieval model (often through score combinations) [78, 82, 14, 15, 104]. A variety of proximity measures were proposed, e.g., minimum term span, minimum pair-wise distance, etc.; in [104], the authors systematically examined different measures and concluded that the minimum pair-wise distance is most effective.

An indirect way to capture proximity in the language modeling framework is to use high-order n-grams as units to represent text. For example, in [98], bigram and trigram language models were shown to outperform simple unigram language models. However, n-gram language models cannot capture dependency of non-adjacent terms (we may attempt to capture such proximity by increasing the length of an n-gram, but it is impractical to enumerate all lengths). A more general way to capture proximity through using appropriate "matching units" is Metzler and Croft's work on term dependency [74]. In that work, term structures with different levels of proximity can be defined in a general probabilistic model. Unfortunately, they only attempted to use a standard language model as a black box to implement the proximity heuristic.

Our work differs from the previous studies in two important aspects. First, we propose a new type of language model that incorporates the term proximity evidence in a model-based approach; thus, the existing language modeling techniques (e.g., mixture-model based feedback [120]) can be applied to our model naturally. Second, we capture term proximity directly based on proximity-based term propagation functions.

In passage retrieval [90, 16, 52, 63, 105], documents are often pre-segmented into small passages, which are then taken as units for retrieval. Also documents can be segmented in a more dynamic way defined at query time, referred to as arbitrary passages [52] ("arbitrary" means that a passage can start at any position in a document). Two subclasses are further defined: fixed-length arbitrary passages resemble overlapped windows but with an arbitrary starting point; variable-length arbitrary passages can be of any length. Fixed-length arbitrary passage retrieval was shown to be as effective as, but more efficient than variable-length arbitrary passages [52]. The proposed PLM covers the fixed-length arbitrary passage as a special case, and can be viewed as a "soft" fixed-length passage. However, different from general passage retrieval, which only models term position evidence indirectly using a standard language model as a black box, our model can incorporate term position information directly into the estimation of language models using a proximity-based propagation function.

Proximity-based density functions have been used to propagate term influence in [26, 56, 73, 80]. Kretser's work [26] proposed to propagate the $tf \cdot idf$ score of each query term to other positions, based on several proximity-based

kernel functions. The document is scored using the position with the highest accumulative $tf \cdot idf$ score finally. But their methods have not been able to achieve effective retrieval performance. The studies [56, 73] are very similar to [26] and based on the vector space model and Boolean model respectively. In our work, we also evaluate the kernel functions proposed in these studies. In addition, we also propose several other density functions that are more effective than theirs. Compared with this previous work, our work also differs in that we use the language modeling framework, and incorporate such density functions into the estimation of language models.

Similar to our work, Petkova and Croft's work [80] proposed a proximity-based document representation for name entities. Their work emphasizes terms of proximity to entities by using a proximity-based density function, which is then used to build description for entities. Our work, however, proposes a positional language model for general document retrieval, and we evaluate the empirical performance of a number of proximity-based density functions systematically.

3.3 Positional Language Models

In this section, we propose a positional language model (PLM) to incorporate term position information into the language model so that we can naturally implement retrieval heuristics such as proximity and passage retrieval.

3.3.1 Estimation of Positional Language Models

In most existing work on language models, a document language model is estimated based on only the counts of words in a document, but not the position of words. The main idea of the PLM is to break this limitation and estimate language models based on the position-dependent counts of words. At a high-level, our idea is to define a language model for each word position in a document. This language model is intended to capture the content of the document at the position, which is roughly like a "fuzzy passage" centered at this position but can potentially cover all the words in the document with less weight on words far away from the position.

Specifically, we assume that a term at each position can *propagate* its occurrence at that position to other positions within the same document through a proximity-based density function, as shown in Figure 3.1. The idea is that if a word w occurs at position i, we would like to pretend that the same word has also occurred at all other positions with a discounted count such that if the position is closer to i, the propagated count for word w at that position would be larger than the propagated count at a position farther away, even though both propagated counts would be less than one, which is the count of w at position i.

The PLM at each position can then be estimated based on all the propagated counts of all the words that to the position as if all the words had appeared actually at the position with discounted counts. Such a PLM intuitively gives

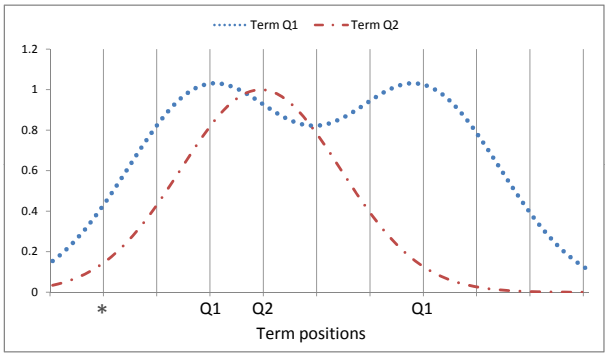

Figure 3.1: Examples of term propagation.

a position-specific view of the content of the document, and thus can naturally support passage retrieval. It can also implement the proximity heuristic because of the use of a proximity-based propagation function.

Once we obtain a PLM for each position in a document, we can use each PLM as a regular document language model for matching with a query. We can then score the document by using one or combining multiple PLMs as we will explain later.

We now present PLM more formally. We first introduce the following notations. Let $D = (w_1, ..., w_i, ..., w_j, ..., w_N)$ be a document, where 1, i, j, and N are absolute positions of the corresponding terms in the document, and obviously N is the length of the document.

- **c(w, i):** the count of term w at position i in document D. If w occurs at position i, it is 1, otherwise 0.
- **k(i, j):** the propagated count to position i from a term at position j (i.e., w_j). Intuitively, given w_j, $k(i, j)$ serves as a discounting factor and can be any non-increasing function of $|i - j|$, that is, $k(i, j)$ favors positions close to j. $k(i, j)$ plays an important role in PLMs, and we will analyze and explore a number of proximity-based density functions.
- **c′(w, i):** the total propagated count of term w at position i from the occurrences of w in all the positions. That is, $c'(w, i) = \sum_{j=1}^{N} c(w, j)k(i, j)$. Thus even if $c(w, i)$ is 0, $c'(w, i)$ may be greater than 0. As shown in Figure 3.1, after propagation, position $'*'$ has a non-zero "count" of terms Q_2 and Q_1.

Based on term propagation, we have a term frequency vector $\langle c'(w_1, i), ..., c'(w_N, i)\rangle$ at position i, forming a *virtual* document D_i. We can see that term position information has been translated to term frequency information stored in this vector. Thus the language model of this virtual document can be estimated as:

$$p(w|D, i) = \frac{c'(w, i)}{\sum_{w' \in V} c'(w', i)} \tag{3.1}$$

where V is the vocabulary set. We call $p(w|D,i)$ a **Positional Language Model** (PLM) at position i.

Intuitively, we can imagine that the PLMs give us multiple representations of D. Thus given a query Q, we can adopt the KL-divergence retrieval model [58] to score each PLM as follows:

$$S(Q,D,i) = -\sum_{w \in V} p(w|Q) \log \frac{p(w|Q)}{p(w|D,i)} \tag{3.2}$$

where $p(w|Q)$ is an estimated query language model. We can estimate $p(w|Q)$ with the maximum likelihood estimate or through some pseudo relevance feedback algorithms (e.g., relevance model [61] or mixture model [120]).

Similar to a regular document language model, the PLM also needs to be smoothed to solve the zero probability problem and to penalize common terms [121]. We consider two popular smoothing methods: Dirichlet prior and Jelinek-Mercer. Dirichlet prior smoothing has proven an effective smoothing method for document language models and captures the document length normalization heuristic [121]. For a PLM, the length of the virtual document at position i is $Z_i = \sum_{w \in V} c'(w,i)$. We use the general collection language model $p(w|\mathcal{C})$ as our background model. Thus the smoothed model is given by:

$$p_\mu(w|D,i) = \frac{c'(w,i) + \mu p(w|\mathcal{C})}{Z_i + \mu} \tag{3.3}$$

where μ is a smoothing parameter. Although previous work has shown that Jelinek-Mercer does not work as well as Dirichlet prior [121], it is unclear whether the same conclusion holds for PLMs because the virtual document length Z_i at different positions are similar to each other [66]. We thus also consider it as an alternative smoothing method, which is given by:

$$p_\lambda(w|D,i) = (1-\lambda) p(w|D,i) + \lambda p(w|\mathcal{C}) \tag{3.4}$$

where λ is a smoothing parameter.

3.3.2 Proximity-based Count Propagation

Clearly, a major technical challenge in PLMs is how to define the propagation function $k(i,j)$. Following some previous work [26, 56, 80], we present here four representative kernel functions: Gaussian, Triangle, Cosine, and Circle, as shown in Figure 3.2. Different kernels lead to different PLMs.

1. Gaussian kernel

$$k(i,j) = \exp\left[\frac{-(i-j)^2}{2\sigma^2}\right] \tag{3.5}$$

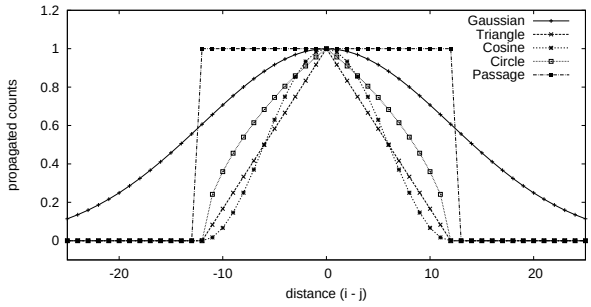

Figure 3.2: Proximity-based kernel functions. We set $\sigma = 12.5$ for all kernels.

2. Triangle kernel

$$k(i,j) = \begin{cases} 1 - \frac{|i-j|}{\sigma} & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

3. Cosine (Hamming) kernel

$$k(i,j) = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\frac{|i-j|\cdot\pi}{\sigma}\right)\right] & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

4. Circle kernel

$$k(i,j) = \begin{cases} \sqrt{1 - \left(\frac{|i-j|}{\sigma}\right)^2} & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

All these four kernels have one parameter σ to tune, which controls the spread of kernel curves, i.e., it restricts the propagation scope of each term. In general, the optimal setting of σ for a term may vary according to the term and may also depend on the query because some general terms presumably would have wider semantic scope in a document, thus requiring a higher value of σ, and similarly, some general query might match a longer relevant passage than a more specific query. Our definition of PLMs would in principle allow us to explore such options. However, as a first study of PLMs, in this chapter, we simply assume that σ is set to the constant across all the terms and all the queries, leaving further optimization of σ as a future work.

As a baseline, we also present the following *non-proximity-based* Passage kernel:

5. Passage kernel:

$$k(i,j) = \begin{cases} 1 & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

With the passage kernel, the PLM can recover the fixed-length arbitrary passage retrieval method [52] in the language

modeling framework. We would use this kernel as a baseline to examine whether the proximity-based kernel functions perform better than this non-proximity-based kernel.

According to the proximity-based propagation property, $p(w|D,i)$ is mainly influenced by terms around the position i, all of which form a "soft" passage together. Hence, the PLM captures a "soft passage" naturally in the language modeling framework. Moreover, different from general passage retrieval, which only captures term position evidence indirectly, our model can measure term position information and incorporate it into a language model directly.

Furthermore, if we set σ to a very large or infinite value for any of the proposed kernels, we would have $k(i,j) = 1$ for all i and j. Thus, we have $c'(w,i) = \sum_{j=1}^{N} c(w,j)k(i,j) = c(w,D)$, which means that $p(w|D,i)$ degenerates to the basic whole document language model $p(w|D)$. This shows that the PLM can cover the basic language model as a special case. In general, we can balance the local term proximity evidence and the document level term statistics by tuning the parameter σ (a small σ would emphasize more on local term proximity). Thus, PLM captures term proximity information in the language modeling framework in a natural way.

3.3.3 Model Implementation

If the PLM is naively implemented, the cost of estimating and ranking PLMs can be extremely high, since the number of positions is much larger than the number of documents or predefined passages. Fortunately, with some mathematical transformation, we may significantly reduce the computational complexity. Below we will show that under reasonable assumptions, PLMs can be implemented similarly to the fixed-length arbitrary passage retrieval.

Given a query, suppose all terms in a document have the same propagation function with the same σ, and the curve of the kernel density function is symmetric. Then we have $k(i,j) = k(j,i)$. Since the most time-consuming part is to compute the normalized length $Z_i = \sum_{w \in V} c'(w,i)$, we rewrite it as:

$$\sum_{w \in V} c'(w,i) \quad = \quad \sum_{w \in V} \sum_{j=1}^{N} c(w,j)k(i,j) = \sum_{j=1}^{N} \left(\sum_{w \in V} c(w,j) \right) k(i,j) = \sum_{j=1}^{N} k(i,j) = \sum_{j=1}^{N} k(j,i)$$

This means the sum of propagated count *to* a position is equal to that propagated *from* the position. We show the computation of Z_i for the Gaussian kernel as an example:

$$\begin{aligned} \sum_{j=1}^{N} k(j,i) &= \sum_{j=1}^{N} \left(exp\left[\frac{-(j-i)^2}{2\sigma^2} \right] \right) \\ &\approx \sqrt{2\pi\sigma^2} \cdot \int_{1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} exp\left[\frac{-(x-i)^2}{2\sigma^2} \right] dx \\ &= \sqrt{2\pi\sigma^2} \cdot \left[\Phi\left(\frac{N-i}{\sigma} \right) - \Phi\left(\frac{1-i}{\sigma} \right) \right] \end{aligned} \tag{3.10}$$

where $\phi(\cdot)$ is the cumulative normal distribution and N is the document length. To calculate $\phi(\cdot)$, we can adopt some existing algorithms, such as the algorithm 26.2.17 [2]. For other kernels considered by us, it is also easy to obtain their cumulative distribution functions through integration.

From this analysis, we can see that our PLM can be implemented similarly to fixed-length arbitrary passage retrieval model. Thus, we can use the techniques proposed in [53] for passage retrieval to implement PLMs; with such an implementation, ranking documents based on PLMs has a complexity of the same order as regular document ranking.

3.4 Document Ranking based on Positional Language Models

As discussed earlier, with PLMs, we can compute a position-specific score $S(Q, D, i)$ for each position i using the KL-divergence of the PLM at the position and the query language model. Such position-specific scores serve as the basis for computing an overall score for document D. We now discuss several different ways of doing this.

3.4.1 Best Position Strategy

Our first strategy is to simply score a document based on the score of its best matching position, formally,

$$S(Q, D) = \max_{i \in [1,N]} \{S(Q, D, i)\} \tag{3.11}$$

This strategy resembles most existing studies on passage retrieval, which generally considered evidences from the best matching passage [16, 52, 63].

3.4.2 Multi-Position Strategy

A more flexible alternative strategy is to first compute the scores of top-k positions separately, and then combine these scores together to take advantage of the evidence from several top ranked positions. Particularly, we can take the average of the top-k scores to score a document:

$$S(Q, D) = \frac{1}{k} \sum_{i \in TopK} S(Q, D, i) \tag{3.12}$$

where $TopK$ is the set of positions corresponding to the top-k highest scores of $S(Q, D, i)$.

	AP88-89	FR	TREC8	WT2G
queries	51-100	51-100	401-450	401-450
#qry(with qrel)	49	21	50	50
mean(ql)	3.70	4.19	2.46	2.46
#total_qrel	4418	502	4728	2279
#documents	164,597	45,820	528,155	247,491
mean(dl)	462	1498	481	1056

Table 3.1: Document set characteristic

3.4.3 Multi-σ Strategy

In this strategy, we compute the best position scores for several different σ values, and then combine these scores together as the final score for a document. The idea is to use different σ values to capture proximity at different propagation ranges.

$$S(Q,D) = \sum_{\sigma \in R} [\beta_\sigma \cdot \max\{S_\sigma(Q,D,i)\}] \tag{3.13}$$

where R is a predefined set of σ values, $S_\sigma(\cdot)$ is the score function for PLMs with parameter σ, β_σ is the weight on different σ ($\sum_{\sigma \in R} \beta_\sigma = 1$). In particular, if $R = \{\sigma_0, \infty\}$, this strategy equals to an interpolation of the PLM (with a parameter σ_0) and the regular document language model. Considering the efficiency issue, we only evaluate this special case of multi-σ strategy, defined formally as follows:

$$S(Q,D) = \gamma \cdot \max_{i \in [1,N]} \{S_{\sigma_0}(Q,D,i)\} + (1-\gamma) \cdot [-D(\theta_Q || \theta_D)] \tag{3.14}$$

3.5 Experiments

3.5.1 Experimental Setup

We used several standard TREC data sets in our study: AP88-89, FR, TREC8, and WT2G. They represent different sizes and genre of text collections. AP88-89 is chosen as a homogeneous collection. FR is selected as a collection of long documents, with a large variance in the document length. TREC8 is a relatively large heterogeneous collection, while WT2G is Web data. Queries are taken from the title field of the TREC topics [1]. Table 3.1 shows some basic statistics about these data sets, including the query topics, number of queries with relevance judgments, mean query length, total number of relevance judgments, number of documents, and mean document length. The preprocessing of documents and queries is minimum, involving only stemming with the Porter stemmer. No stop words have been removed.

In each experiment, we first use the baseline model (KL-divergence) to retrieve $2,000$ documents for each query,

[1] Topic 110 in AP88-89 was left out accidently due to a format problem in preprocessing.

WT2G					
kernel$\diagdown\sigma$	25	75	125	175	275
Gaussian	**0.2989**	**0.3213**	**0.3286**	**0.3307**	0.3285
Triangle	0.2661	0.3028	0.3149	0.3211	**0.3288**
Cosine	0.2621	0.3007	0.3128	0.3181	0.3243
Circle	0.2797	0.3140	0.3225	0.3273	0.3267

FR					
Gaussian	**0.2913**	0.2679	0.2895	0.2880	0.2846
Triangle	0.2585	0.2898	0.2858	0.2682	**0.2897**
Cosine	0.2603	**0.2910**	**0.3000**	**0.2948**	0.2858
Circle	0.2685	0.2754	0.2673	0.2877	0.2873

TREC8					
kernel$\diagdown\sigma$	25	75	125	175	275
Gaussian	**0.2364**	**0.2465**	**0.2503**	**0.2535**	**0.2550**
Triangle	0.2244	0.2379	0.2438	0.2475	0.2500
Cosine	0.2257	0.2390	0.2430	0.2457	0.2486
Circle	0.2315	0.2401	0.2464	0.2492	0.2523

AP88-89					
Gaussian	**0.1926**	**0.2112**	**0.2162**	**0.2177**	**0.2198**
Triangle	0.1709	0.1987	0.2077	0.2117	0.2173
Cosine	0.1682	0.1969	0.2063	0.2107	0.2144
Circle	0.1801	0.2034	0.2093	0.2135	0.2159

Table 3.2: MAP Results of different kernel functions with Dirichlet smoothing method.

and then use the PLM (or a baseline method) to re-rank them. The top-ranked $1,000$ documents for all runs are compared using the mean average precisions (MAP) as the main metric.

3.5.2 Best Position Strategy

We first examine the effectiveness of the Best Position Strategy for scoring documents based on PLM. Since the performance of this strategy is directly determined by the effectiveness of the kernel function used to estimate the PLM, we first compare the proposed four different proximity-based kernel functions to see which one performs the best. For this comparison, the initial retrieval results were obtained using the KL-divergence retrieval model with Dirichlet prior smoothing; since the *relative* performance of different kernel functions would presumably not be affected by the setting of the smoothing parameter in the initial retrieval, we did not tune the smoothing parameter and simply set it to $1,000$. To compare different kernel functions, we follow [52] and systematically test a set of fixed σ values from 25 to 300 in increments of 25. For the sake of efficiency, positions start at 25-word intervals, which was shown by [51] to be an effective way for passage retrieval.

Since we also smooth an estimated PLM when computing retrieval scores, we test both Dirichlet prior smoothing

WT2G					
kernel$\diagdown\sigma$	25	75	125	175	275
Gaussian	**0.3024**	**0.3170**	0.3133	0.3096	0.3010
Triangle	0.2711	0.3057	0.3118	0.3170	0.3131
Cosine	0.2622	0.2855	0.2681	0.2452	0.2039
Circle	0.2813	0.3130	**0.3188**	**0.3179**	**0.3148**

FR					
Gaussian	**0.2639**	0.2606	0.2592	**0.2827**	0.2822
Triangle	0.2458	**0.2681**	0.2607	0.2610	**0.2834**
Cosine	0.2463	0.2476	0.2424	0.2249	0.1593
Circle	0.2512	0.2557	**0.2613**	0.2591	0.2833

TREC8					
kernel$\diagdown\sigma$	25	75	125	175	275
Gaussian	**0.2454**	**0.2510**	**0.2548**	**0.2575**	**0.2576**
Triangle	0.2335	0.2477	0.2491	0.2506	0.2562
Cosine	0.2335	0.2423	0.2356	0.2227	0.2058
Circle	0.2369	0.2456	0.2498	0.2528	0.2555

AP88-89					
Gaussian	**0.1892**	**0.2016**	**0.2054**	**0.2066**	0.2049
Triangle	0.1718	0.1933	0.1968	0.2002	**0.2051**
Cosine	0.1701	0.1910	0.1815	0.1636	0.1349
Circle	0.1735	0.1933	0.1962	0.2010	0.2049

Table 3.3: MAP Results of different kernel functions with Jelinek-Mercer smoothing method

(with parameter $1,000$) and Jelinek-Mercer (with parameter 0.5). (see Equations 3.3 and 3.4). The results of comparing different kernel functions when using each smoothing method are shown in Table 3.2 and Table 3.3, respectively. The best result for each σ value is highlighted. Overall, we see that for all kernels, a relatively large σ value, e.g., 125, 175, and 275, often brings the best performance. It seems that the performance of all runs stabilizes after σ reaches 125. Considering the length of the soft passage is approximately 2σ (as shown in Figure 3.2), this result confirms the observation in recent studies of passage retrieval [52, 63] that setting passage length to a value around 350 often achieves the best performance. Among all the kernel functions, the Gaussian kernel clearly has the best performance; it contributed 15 best MAP scores out of 20 for Dirichlet prior smoothing and 13 out of 20 for Jelinek-Mercer. To see whether the setting of the smoothing parameter may have affected the relative performance of these kernel functions, we further compare them for a wide range of values of the Dirichlet prior parameter on TREC8 in Figure 3.3, where we fix $\sigma = 175$ for all kernels. The results clearly show that the Gaussian kernel is the winner among all the four functions. One way to explain why the Gaussian kernel performs the best is that it is the only one of all the functions that exhibits the following property: the propagated count would drop slowly when the distance value $|i - j|$ is small, but drop quickly as the distance value is in a middle range, and then drop slowly again when distance value becomes very large. Such an "S-shape" trend is reasonable for the following reason. Dependent terms are not always adjacent

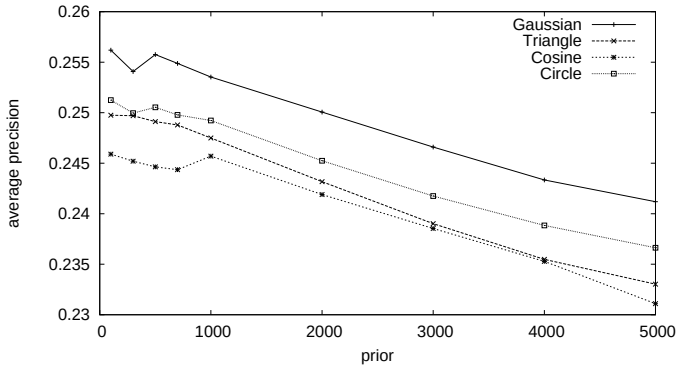

Figure 3.3: Sensitivity to Dirichlet smoothing parameter of different kernels over TREC8

in documents, but can be a little far from each other, thus we would not like to make the propagation so sensitive to the distance when the distance is small. However, when the distance is just around the *boundary* of strong semantic associations (semantic scope of a term), the propagated count should be more sensitive to the distance change. Then as the distance increases further, all terms are presumably only loosely associated, and thus the propagated term count again should not be so sensitive to the difference of distances.

Since the Gaussian kernel performs the best, in all the following experiments, we use this kernel function. In order to see whether PLM can effectively capture proximity and passage retrieval, we compare the performance of Gaussian kernel ($\sigma = 175$) with the baseline whole document language model using both Dirichlet prior smoothing and Jelinek-Mercer smoothing (both the PLM and the baseline would use the same smoothing method). The results are shown in Figure 3.4, where we vary the smoothing parameters for both smoothing methods on all the four data sets. We can observe that the PLM improves performance on WT2G and FR clearly and consistently, which shows that, similar to general passage retrieval, the PLM can bring added benefits to document retrieval when documents are relatively long. Some improvements are also found on TREC8, possibly because it is a heterogeneous data set as compared to AP88-89, which is homogeneous; a heterogeneous collection is relatively nosier, thus term dependence information may be more helpful. Unfortunately, the PLM does not seem to show its advantages on AP88-89 for Dirichlet prior smoothing even though it is so for Jelinek-Mercer smoothing.

By comparing the results of the two smoothing methods in Figure 3.4, we see that in general, Dirichlet prior performs better than Jelinek-Mercer for PLM, and the Dirichlet prior smoothing method seems to perform stably for a range of μ values around 500.

Figure 3.4 shows that PLM outperforms whole document LM baseline likely due to the use of proximity and passage-retrieval heuristics. We would like to further understand whether PLM, which captures "soft" passages, is also better than the fixed-length arbitrary passage retrieval method. Thus, we compare the PLM (using the Gaussian kernel, $\sigma = 175$, Dirichlet prior smoothing, $\mu = 500$) with the fixed-length arbitrary passage retrieval method [52]

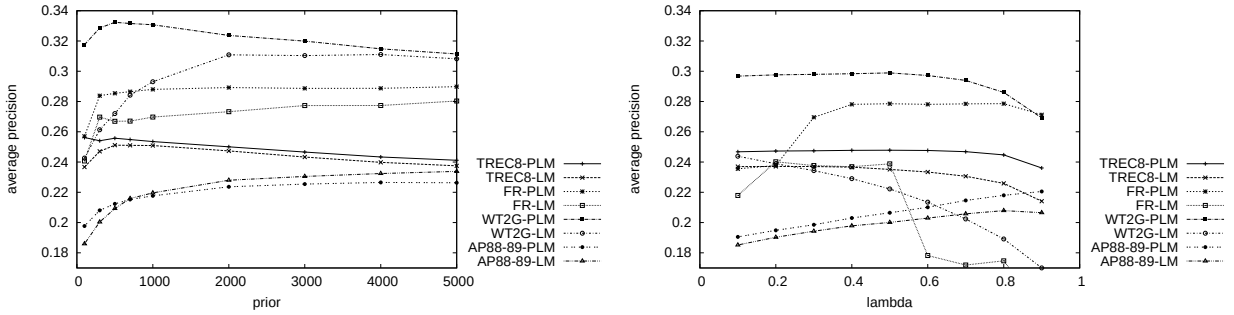

Figure 3.4: Sensitivity to the smoothing parameters of Dirichlet smoothing (left) and Jelinek-Mercer smoothing (right) of basic language modeling methods (LM) and the PLM on four collections.

WT2G					
–	25	75	125	175	275
Psg	0.2962	0.3176	0.3225	0.3265	0.3249
PLM	**0.2989**	**0.3213**	**0.3286**$^+$	**0.3307**$^+$	**0.3285**
TREC8					
Psg	0.2358	0.2433	0.2492	0.2511	0.2518
PLM	**0.2364**	**0.2465**$^+$	**0.2503**	**0.2535**$^+$	**0.2550**$^+$
FR					
Psg	0.2899	**0.2704**	0.2878	**0.2887**	**0.2860**
PLM	**0.2913**	0.2679	**0.2895**	0.2880	0.2846
AP88-89					
Psg	0.1854	0.2054	0.2130	0.2142	0.2154
PLM	**0.1926**$^+$	**0.2112**$^+$	**0.2162**$^+$	**0.2177**$^+$	**0.2198**$^+$

Table 3.4: Comparison of fixed-length arbitrary passage retrieval (Psg) and PLM. $'+'$ means that improvements over the Psg are statistically significant.

(i.e., the Passage kernel). The MAP scores are summarized in Table 3.4, where the best result for each σ is highlighted. We can observe that the PLM indeed outperforms the standard passage retrieval baseline significantly, which shows that modeling term proximity directly using a proximity-based density function is more effective and robust than assuming fixed lengths.

3.5.3 Multi-position Strategy

We now evaluate the multi-position ranking strategy. Based on the observation in the previous section, we use the Gaussian kernel ($\sigma = 175$) with Dirichlet smoothing ($\mu = 500$) for the PLM. We vary parameter k and plot the MAP results of multi-position strategy in Figure 3.5. We see that the multi-position strategy does not lead to any noticeable improvement over the best position strategy (i.e., $k = 1$), and if we use a relatively large k, the performance can even degrade dramatically. Hence, given a single σ value, the best position strategy is a robust and reasonable method for document ranking.

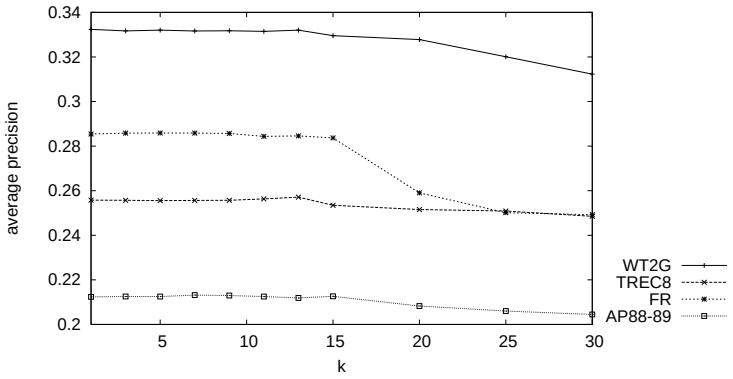

Figure 3.5: Sensitivity to the parameter k of multi-position strategy

method\data	WT2G	TREC8	FR	AP88-89
KL	0.2931	0.2509	0.2697	0.2196
$\sigma = 25$	0.3247^{+}	0.2562^{+}	**0.2936**	**0.2237**$^{+}$
$\sigma = 75$	**0.3336**$^{+}$	0.2553^{+}	0.2896^{+}	0.2227
$\sigma = 125$	0.3330^{+}	0.2559^{+}	0.2885	0.2201
$\sigma = 175$	0.3324^{+}	**0.2574**$^{+}$	0.2858	0.2196
$\sigma = 275$	0.3255^{+}	0.2561^{+}	0.2852	0.2193

Table 3.5: The best performance of multi-σ strategy for different σ. $'+'$ means that improvements over the baseline KL method are statistically significant.

3.5.4 Multi-σ Strategy

We now turn to the evaluation of the multi-σ strategy – in particular, the special case of interpolating a PLM with the whole document language model (i.e., $\sigma = \infty$). To test this special case of Multi-σ strategy, we fix one σ value to ∞, and vary the other one from 25 to 300 in increments of 25. For each σ value, we again use the Gaussian kernel and the Dirichlet prior smoothing method ($\mu = 500$). The results are presented in Table 3.5, where we tune the interpolation coefficient γ in the range of $[0.0, 1.0]$ to its optimal value for each σ. It shows that, when interpolated with document language models, the PLM performs more robustly and effectively. One possible explanation is that a locally focused PLM alone does not model document-level retrieval heuristics as effectively as the whole document language model does, even though the former captures term proximity heuristic better, thus balancing them will get better results. Another interesting observation is that the best results are always obtained when we use a smaller σ value, e.g. 25 or 75, which also suggests that the PLM is better at capturing local term proximity evidence rather than document-level evidence (e.g., term frequency).

To further look into the sensitivity to γ, we set $R = \{75, \infty\}$ and vary γ on all the four data sets. The results are shown in Figure 3.6. Interestingly, for collections of long documents (i.e., FR and WT2G), we can rely more on PLMs (larger γ), likely because the whole document language models may contain much noise, but for collections of short-documents (i.e., TREC8 and AP88-89), the sensitivity curves are generally flatter and a relatively smaller γ seems

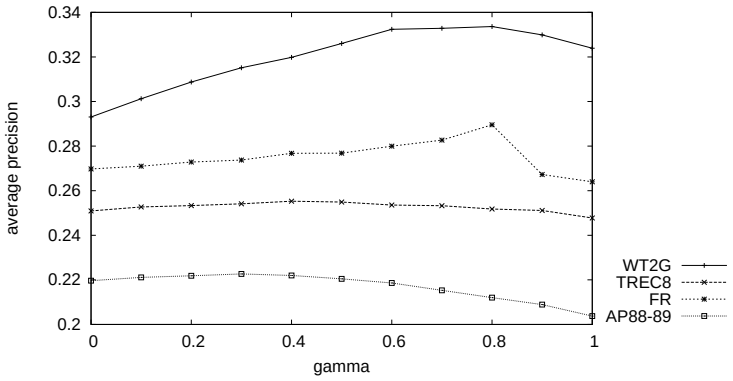

Figure 3.6: Sensitivity to γ value of multi-σ strategy.

method\data	WT2G	TREC8	FR	AP88-89
R_1 + MinDist	0.3197	**0.2568**	0.2708	0.2220
$R = \{75, \infty\}$	**0.3336**	0.2553	**0.2896**	**0.2227**

Table 3.6: MAP Comparison of the multi-σ strategy and the best method proposed by Tao and Zhai.

working better, suggesting that regular document language models work reasonably well without term proximity information.

We finally compare our multi-σ strategy ($R = \{75, \infty\}$, $\gamma = 0.8$ for FR and WT2G, and $\gamma = 0.4$ for TREC8 and AP88-89) with a state-of-the-art proximity retrieval method proposed in [104]. Our parameter setting gives PLM near optimal performance. To be fair, we also use the best language modeling retrieval formula suggested in [104] (i.e., R_1 + MinDist), and tune their parameter α to its optimal value to re-rank documents. We label this run as "R_1 + MinDist" and report the comparison results in Table 3.6. Interestingly, we see both methods perform similarly on short-document collections (i.e., TREC8 and AP88-89), but our method is clearly better on long-document collections (i.e., WT2G and FR), suggesting that the proposed PLM can capture the passage and proximity heuristics more effectively.

3.6 Summary

In this chapter, we proposed a novel positional language model which implements both proximity heuristic and passage retrieval in a unified language model. We proposed and studied four different proximity-based density functions to estimate PLMs. Experiment results show that the Gaussian density kernel performs the best, followed by Circle, Triangle, and Cosine. As for the smoothing of PLM, the Dirichlet smoothing method performs better than Jelinek-Mercer smoothing.

In addition, we further proposed three PLM-based document ranking strategies. We evaluated their performance and found that the multi-σ strategy performs the best. Our experiments on several standard test collections show that the proposed PLM not only outperforms the regular document language models, but also outperforms the fixed-length

arbitrary passage retrieval method and a state-of-the-art proximity-based retrieval model.

As a new family of language models, the PLM opens up many interesting future research directions. One of the most interesting directions is to further study whether setting a term-specific and/or query-specific σ can further improve performance. Another interesting direction is to study how to optimize σ automatically based on statistics such as IDF of terms and discourse structures of documents.

Chapter 4

Positional Relevance Models

4.1 Introduction

The previous chapter aims at bridging one theory-effectiveness gap in *document* language models by making them be able to capture term position and proximity heuristics. Accurate estimation of *query* language models also plays a critical role in the language modeling approaches. The most effective methods for estimating query language models generally rely on the strategy of pseudo-relevance feedback (PRF) [88, 85, 91, 11, 87, 61, 120], which can improve retrieval performance significantly over simple estimation methods that only use the query [69], as we have discussed in Section 2.5.

However, existing PRF algorithms use a whole feedback document as a unit for selecting expansion terms, and simply assume that all terms in a feedback document are equally useful, again ingoring term occurrence positions and the term proximity evidence. This is often non-optimal, as a feedback document may cover multiple incoherent topics and thus may contain much irrelevant information as often happens in Web search. The existence of multiple topics and irrelevant information would lead to a noisy feedback model as potentially harmful terms from non-relevant topics may be picked up to include in the feedback model. As a result, the use of pseudo feedback may not improve or even decrease the retrieval performance. Thus a critical challenge in improving all feedback methods is to effectively select from feedback documents those terms that are most likely relevant to the query topic.

In this chapter, we solve this challenge by exploiting the position and proximity information of terms as cues to assess if a term is related to the query topic. Since topically related content is usually grouped together in text documents, terms closer to the occurrences of query words are, in general, more likely relevant to the query topic, thus a good feedback model should intuitively place higher weights on such terms.

Based on this intuition, we propose a novel positional relevance model (PRM) to incorporate the cues of term positions and term proximity in a probabilistic feedback model based on statistical language modeling. The key idea is to extend the relevance model [61] to aggregate the associations between a term and query words at the position level via the positional language model (PLM) proposed in the previous Chapter 3. An important advantage of estimating a relevance model based on PLM is that it can model the "relevant positions" in a feedback document with probabilistic

models so as to assign more weights to terms at more relevant positions in a principled way, thus leading naturally to selection of expansion terms more likely relevant to the query topic.

Since PRM estimates a relevance model at the level of term positions, it incorporates individual term positions *directly* into a probabilistic model. This is in contrast with virtually all the existing pseudo feedback techniques which have only made use of term statistics at the document level [88, 85, 91, 11, 87, 61, 120, 69], or at the best, at the level of passages [3, 112, 63, 75] without distinguishing every different position.

Analogously to the two methods proposed for estimating the relevance model [61], we also derive two methods for estimating PRM, leading to two different ways to aggregate term information based on positions. We evaluate the proposed PRM on two large TREC datasets. Experimental results demonstrate that PRM is effective in exploiting term proximity for pseudo feedback and significantly outperforms the relevance model in both document-based feedback and passage-based feedback.

4.2 Related Work

General Pseudo-Relevance Feedback: Pseudo-relevance feedback has been shown to be effective with various retrieval models [88, 85, 91, 11, 87, 61, 120, 69]. In the vector space model, feedback is usually done by using the Rocchio algorithm, which forms a new query vector by maximizing its similarity to pseudo-relevant documents [88]. The feedback method in classical probabilistic models is to select expansion terms primarily based on Robertson/Sparck-Jones weight [85]. Several query expansion techniques have been developed in the language modeling framework, including, e.g., the mixture-model feedback method [120] and the relevance model [61], which have been discussed in Chapter 2. The basic idea is to use feedback documents to estimate a better query language model. Both the mixture model and relevance model have been shown to be very effective, but the relevance model appears to be more robust [69].

All these pseudo feedback algorithms use a whole feedback document as a unit, and thus term position and proximity evidences are largely ignored. Our work is an extension of the relevance model to estimate a feedback model based on individual term positions.

Passage Feedback: There have been several studies to exploit passage-level evidence of documents for feedback, e.g., [3, 112, 63], which can potentially address the heterogeneous topical structure of documents to some degree. However, these approaches usually take a traditional feedback model as a *black box* to handle sub-document units as if they were regular documents. For example, Liu and Croft's work [63] estimates a relevance model based on the best matching passage of each feedback document, where fixed-length arbitrary passages that resemble overlapped windows but with an arbitrary starting point [52] can often be used due to its effectiveness and efficiency [52, 63]. A

limitation of this approach is that term positions are not directly incorporated into the feedback model. As we will show later in this chapter, the proposed PRM outperforms such a passage feedback approach.

Some other approaches, e.g., [116, 13], make use of visual cues or eye tracker to improve passage feedback for web search: on the server side of a search engine, documents can be decomposed into topically different components via visual cues [116], while on the client side of users, gaze-based attention feedback [13] can go down to the sub-document level by exploiting evidence about which document parts the user looks at. However, such approaches face the same problems as general passage feedback without being able to model each individual position.

In fact, these passage-based or sub-document level feedback models are orthogonal to the proposed PRM in the sense that PRM can be applied to passages to model proximity inside a passages in the same way as it can be applied to whole documents. Moreover, the underlying positional language model, which can capture passage-level evidence in a *soft* way in model estimation, has been shown to work better than imposing a "hard" boundary of passages.

Recently, Metzler and Croft's work on Latent Concept Expansion [75]. Their work provides a more general framework, but is only be able to *indirectly* captures term position and proximity evidence through the use of appropriate general passages. And also their work is complementary with our ideas in that we can use PRM as a more effective feature than the general passage features defined on their graph, so our PRM scores can then be combined with other features explored in [75] to further improve its performance.

Term Proximity in Pseudo-Relevance Feedback: There has been relatively little work done in the area of formally modeling term proximity heuristic in the context of pseudo feedback. However, there have been several attempts to simply combine term proximity with other feedback heuristics to select good expansion terms. In [108], several distance functions were evaluated for selecting query expansion terms from windows or passages surrounding query term occurrences; however, no improvement was observed as compared to existing feedback methods. Cao et al. [17] used a supervised method to classify whether an individual expansion term is good or not, in which term proximity is one of their features. Their method only loosely combined term proximity with traditional feedback heuristics; in contrast, we incorporate term position and proximity into a probabilistic feedback model with more meaningful parameters.

4.3 Positional Relevance Model

In this section, we describe the proposed positional relevance model (PRM) which incorporates term position information into the estimation of feedback models so that we can naturally reward terms close to query terms in the feedback documents and avoid including irrelevant terms in the feedback model.

The proposed PRM can be regarded as an extension to the relevance model (RM) [61], which have been reviewed in Chapter 2.

In the relevance model, the count of a term is computed over an entire feedback document. The main idea of the proposed positional relevance model (PRM) is to further distinguish different positions of a term and discount the occurrences of a term at positions that are far away from a query term in a feedback document.

Similarly to RM, a PRM is also a multinomial distribution $P(w|Q)$ that attempts to capture the probability that term w is seen in a relevant document. However, PRM goes beyond RM to estimate the conditional probability $P(w|Q)$ in terms of the joint probability of observing w with the query Q at every position in every feedback document. Formally,

$$P(w|Q) = \frac{P(w,Q)}{P(Q)} \propto P(w,Q) = \sum_{D \in \mathcal{F}} \sum_{i=1}^{|D|} P(w,Q,D,i) \tag{4.1}$$

where i indicates a position in document D, and F is the set of feedback documents (assumed to be relevant).

The challenge now lies in estimating the joint probability $P(w,Q,D,i)$. Inspired by the two estimation methods proposed in [61] for estimating relevance models, we derive two methods similarly for estimating $P(w,Q,D,i)$. The first method assumes that w is sampled in the same way as Q based on a query generation process, while the second method assumes that w and Q are sampled using two different mechanisms based on a document generation process.

4.3.1 Estimation Method 1: Query Generation

In this method, we first compute the joint probability of observing a word together with the query words at each position and then aggregate the evidence by summing over all the possible positions. Specifically, we factor the joint probability $P(w,Q,D,i)$ for each pseudo-relevant document D as follows:

$$P(w,Q,D,i) = P(D)P(i|D)P(w,Q|D,i) \tag{4.2}$$

Intuitively, we have assumed a generative model in which we would first pick a document according to $P(D)$, then choose a position i in document D with probability $P(i|D)$, and finally generate word w and query Q conditioned on D and i, with probability $P(w,Q|D,i)$.

$P(D)$ can be interpreted as a document prior and set to a uniform distribution with no prior knowledge about document D. While it is possible to estimate $P(i|D)$ based on document structures, here we assume that every position is equally likely, i.e., $P(i|D) = \frac{1}{|D|}$. Improving the estimation of $p(D)$ and $P(i|D)$ would be an interesting future work. An illustration of the dependencies between the variables involved in the derivation is shown on Figure

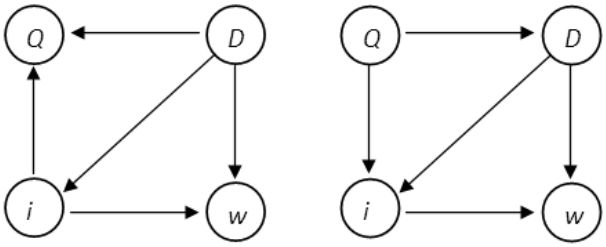

Figure 4.1: Dependence networks for two methods, i.e., method 1 (left) and method 2 (right), of estimating positional relevance models.

4.1 (left side).

After making these assumptions and a further assumption that the generation of word w and that of query Q are independent, we have

$$P(w, Q, D, i) \propto \frac{P(w, Q|D, i)}{|D|} = \frac{P(Q|D, i)P(w|D, i)}{|D|} \tag{4.3}$$

Plugging Equation 4.3 into Equation 4.1, we obtain the following estimate of the PRM:

$$P(w|Q) \propto P(w, Q) \propto \sum_{D \in \mathcal{F}} \sum_{i=1}^{|D|} \frac{P(Q|D, i)P(w|D, i)}{|D|} \tag{4.4}$$

In the above equation, $P(w|D, i)$ is the probability of sampling word w at position i in document D. To improve the efficiency of PRM, we simplify $P(w|D, i)$ as:

$$P(w|D, i) = \begin{cases} 1.0 & \text{if } w \text{ occurs at position } i \text{ in } D \\ 0.0 & \text{otherwise} \end{cases} \tag{4.5}$$

The term $P(Q|D, i)$ in Equation 4.4 is the key component in estimating the positional relevance model. It is the query likelihood at position i of document D, and we will discuss how to estimate it based on the positional language model in Section 4.3.4. Additionally, there is a third term $|D|$ in the equation, which penalizes long documents to prevent them from dominating the feedback model (long documents naturally have more positions).

Thus, Equation 4.4 essentially combines all terms in feedback documents by assigning different weights to each term: (1) $P(Q|D, i)$ serves as a relevance-based weight for each position in each document so that a position with many query terms nearby would have a higher weight. Thus as an intra-document weight, $P(Q|D, i)$ can measure the relative weights of positions within a document: a position closer to query words would more likely generate the query, and as a result a term that occurs at this position would naturally receive a higher weight. (2) $|D|$ comes into

the formula because of the assumption about uniform distribution over all the positions in a document and can be interpreted as an inter-document weight: it penalizes a long document which is reasonable since a longer document by nature has more positions and more occurrences of terms to contribute.

4.3.2 Estimation Method 2: Document Generation

In this method, we consider the following different way to decompose the joint probability distribution:

$$P(w, Q, D, i) = P(Q)P(D|Q)P(i|Q, D)P(w|D, i) \tag{4.6}$$

The assumed generative model is as follows. We first pick a query according to some prior $P(Q)$. We then generate a document D with probability $P(D|Q)$. Finally, we select a position i in D with probability $P(i|Q, D)$ and generate word w according to $P(w|D, i)$. An illustration of this sampling process is given on the right side of Figure 4.1.

For the purpose of estimating $P(w|Q)$, we can clearly ignore the term $P(Q)$ as it is a query-specific constant. Using Bayes Rule and assuming both $P(D)$ and $P(i|D)$ to be uniform (as we have assumed in the first estimation method), we have

$$P(D|Q) = \frac{P(Q|D)P(D)}{\sum_{D\in F} P(Q|D)P(D)} = \frac{P(Q|D)}{\sum_{D\in F} P(Q|D)} \tag{4.7}$$

$$P(i|D, Q) = \frac{P(Q|D, i)P(i|D)}{\sum_{i=1}^{|D|} P(Q|D, i)P(i|D)} = \frac{P(Q|D, i)}{\sum_{i=1}^{|D|} P(Q|D, i)} \tag{4.8}$$

Plugging Equations 4.6, 4.7, and 4.8 into Equation 4.1, we obtain the following estimate of PRM:

$$P(w|Q) \propto \sum_{D\in\mathcal{F}} \sum_{i=1}^{|D|} \frac{P(Q|D)}{\sum_{D\in F} P(Q|D)} \frac{P(Q|D, i)}{\sum_{i=1}^{|D|} P(Q|D, i)} P(w|D, i) \tag{4.9}$$

where $P(Q|D)$ is the query likelihood score of document D, which can be computed using either the positional language models or the standard document language model. In our experiments, we use the latter, i.e., $P(Q|D) = \prod_{j=1}^{m} P(q_j|D)$.

As in the first estimation method, we compute $P(w|D, i)$ using Equation 4.5.

Similarly to the first estimation method, this second estimate of PRM also essentially combines all terms in feedback documents by assigning different weights to each term: the first weighting term in Equation 4.9 is seen to be the normalized query likelihood score of the document, which assigns more weights to documents that are more likely to be relevant, while the second weighting term is the normalized query likelihood of each *positional* language model,

which assigns more weights to terms that are closer to query words.

Compared to the first estimation method, the document length normalizer $|D|$ is missing, but a comparable effect is now achieved by normalizing the query likelihood of each positional language model $P(Q|D,i)$. Indeed, the effect of intra-document weighting and inter-document weighting can now be seen even more clearly, i.e., the normalized $P(Q|D)$ can be interpreted as the inter-document weight favoring a document matching the query well, while the normalized $P(Q|D,i)$ clearly achieves intra-document weighting to place more weight on terms closer to query terms in document D.

4.3.3 Comparison

Comparing the two methods, PRM1 essentially takes each position as a "pseudo document" and applies the general relevance model to such "pseudo documents", while PRM2 extends the general relevance model by re-weighting terms within the same document according to their distances to the occurrences of query terms. PRM1 is more aggressive in that, it directly extracts *absolutely* "relevant" positions from multiple feedback documents, but PRM2 first extracts *relatively* "relevant" positions within each document and then pools these positions together.

To further analyze the technical differences between PRM1 and PRM2, we re-write PRM1 (Formula 4.4) as below:

$$\begin{aligned} P_{prm1}(w|Q) &\propto \sum_{D\in\mathcal{F}}\sum_{i=1}^{|D|}\frac{P(Q|D,i)P(w|D,i)}{|D|} \\ &= \sum_{D\in\mathcal{F}}\sum_{i=1}^{|D|}\frac{\sum_{i=1}^{|D|}P(Q|D,i)}{|D|}\frac{P(Q|D,i)}{\sum_{i=1}^{|D|}P(Q|D,i)}P(w|D,i) \end{aligned} \tag{4.10}$$

Comparing Formula 4.9 and 4.10, we can see that, the ways of assigning weights to each term (position) in PRM1 and PRM2 are essentially the same, both of which use a normalized positional query likelihood score, i.e., $\frac{P(Q|D,i)}{\sum_{i=1}^{|D|}P(Q|D,i)}$, but their strategies for document weighting are quite different: PRM1 uses an *average* of all positional query likelihood scores, i.e., $\frac{\sum_{i=1}^{|D|}P(Q|D,i)}{|D|}$, as the document weight, while PRM2 uses a normalized query likelihood score, i.e., $\frac{P(Q|D)}{\sum_{D\in F}P(Q|D)}$.

4.3.4 More Estimation Details

This section provides the final estimation details for our positional relevance model (Equation 4.4 and 4.9), i.e., how to estimate $P(Q|D,i)$. We adapt the proposed positional language model in Chapter 3 to do that. The PLM at position i of document D can be estimated as:

$$P(w|D,i) = \frac{c'(w,i)}{\sum_{w'\in V}c'(w',i)} \tag{4.11}$$

where $c'(w,i)$ is the total propagated count of term w at position i from the occurrences of w in all the positions. We estimate $c'(w,i)$ using the Gaussian kernel function due to its effectiveness:

$$c'(w,i) = \sum_{j=1}^{|D|} c(w,j) \exp\left[\frac{-(i-j)^2}{2\sigma^2}\right] \tag{4.12}$$

where i and j are absolute positions of the corresponding terms in the document, and $|D|$ is the length of the document; $c(w,j)$ is the *real* count of term w at position j. With the proposed approximation method, the following estimation of $P(w|D,i)$ is obtained:

$$P(w|D,i) = \frac{c'(w,i)}{\sqrt{2\pi\sigma^2} \cdot \left[\Phi\left(\frac{|D|-i}{\sigma}\right) - \Phi\left(\frac{1-i}{\sigma}\right)\right]} \tag{4.13}$$

where $\Phi(\cdot)$ is the cumulative normal distribution and the denominator is essentially the length of the "soft" passage centered at position i.

However, there is one issue with the above estimation: the length of "soft" passages around the *boundaries* of a document would be smaller than that in the *middle* of the document; as a result, boundary positions tend to unfairly receive more weights. This may not raise problems in PLMs for retrieval, but it is a more serious concern for PRM, where the relative weights of terms are more important. So we decide to use a fixed length for all "soft" passages in feedback documents to estimate their corresponding positional language models as follows:

$$P(w|D,i) = \frac{c'(w,i)}{\sqrt{2\pi\sigma^2}} \tag{4.14}$$

This strategy has shown to be better than the original implementation for estimating PRM.

The distribution $P(\cdot|D,i)$ needs to be smoothed. Now that all "soft" passages have equal length, we use Jelinek-Mercer smoothing method to smooth PLM, which is shown to work as well as the Dirichlet prior smoothing method and is relatively insensitive to the setting of σ in our experiments.

$$P_\lambda(w|D,i) = (1-\lambda)P(w|D,i) + \lambda P(w|C) \tag{4.15}$$

where $\lambda \in [0,1]$ is a smoothing parameter and $p(w|C)$ is the collection language model. Now we can compute the positional query likelihood score $P(Q|D,i)$ for position i.

$$P(Q|D,i) = \prod_{j=1}^{m} P_\lambda(q_j|D,i) \tag{4.16}$$

	Terabyte05	Terabyte06	ClueWeb09 Cat. B
queries	751-800	801-850	20001-21000
#qry(with qrel)	50	49	358
#documents	$25,205,179$		$50,220,423$
mean(dl)	931		875

Table 4.1: Document set characteristic

Plugging Equation 4.16 into Equations 4.4 and 4.9, we would be able to compute the two estimation methods directly. Interestingly, if we set $\lambda = 1$ or $\sigma = \infty$, Method 2 will degenerate to the general relevance model (see Equation 2.16).

The computation of positional query likelihood is the most time-consuming part in estimating PRM. Fortunately, there is no serious efficiency concern even with an unoptimized implementation. The reason is because we only need to traverse each position of a document twice: during the first pass, the positions of query terms are recorded; in the second, we compute a positional query likelihood for each position directly based on the position information of query terms collected in the first pass. Therefore, the efficiency is comparable to the estimation of the relevance model.

Finally, the estimated positional relevance model $P(w|Q)$ will also be interpolated with the original query model θ_Q using Equation 2.20 to improve performance with a similar parameter α to that used in the mixture-model feedback [120] and RM3 [1].

4.4 Experiments

4.4.1 Experimental Setup

We used two standard TREC datasets in our study: Terabyte (i.e., the Gov2 collection) and ClueWeb09 Category B. They represent two very large web text collections in English. Queries were taken from the title field of the TREC topics. We used the Lemur toolkit (version 4.10) and Indri search engine (version 2.10) [1] to implement our algorithms. For both datasets, the preprocessing of documents and queries included stemming with the Porter stemmer and stopwords removing using a total of 418 stopwords from the standard InQuery stoplist. Table 4.1 shows some basic statistics about the datasets.

We evaluated seven methods. (1) The basic retrieval model is the KL-divergence retrieval model [58], and we chose the Dirichlet smoothing method [121] for smoothing document language models, where the smoothing parameter μ was set empirically to 1500. This method was labeled as "NoFB". (2) The baseline pseudo feedback method is the relevance model "RM3" described in Section 2.5.1 [1], which is one of the most effective and robust pseudo feedback methods under language modeling framework [69]. (3) Another baseline pseudo feedback method is a standard

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

passage-based feedback model, labeled as "RM3-p", which estimates the RM3 relevance model based on the best matching passage of each feedback document [63]. (4) We have two variations of PRM, i.e., "PRM1" and "PRM2", which are based on the two estimation methods described in Section 4.3, respectively. (5) In addition, we also used PRM1 and PRM2 for passage feedback in a way as RM3-p does. Specifically, we first computed a PLM for each position of the document, and then we estimate a PRM based on a passage of size 2σ centered at the position with the maximum positional query likelihood score (see Equation 4.16). These two runs are labeled as "PRM1-p" and "PRM2-p" respectively.

There are several parameters in these pseudo feedback algorithms. We fixed the number of feedback documents to 20 and the number of terms in feedback model to 30. Other parameters, including the feedback interpolation coefficient α, the two additional parameters σ and λ in PRM, the passage size, and the passage smoothing parameter in RM3-p, were all tuned on Terabyte05 dataset.

We used Terabyte06 and ClueWeb09 for testing. The top-ranked 1000 documents for all runs were compared in terms of their mean average precisions (MAP) (for Terabyte06) or eMAP [19] (for ClueWeb09). In addition, other performance measures, such as Pr@10, Pr@30 and Pr@100 for Terabyte06 and eP@10, eP@30 and eP@100 for ClueWeb09, were also considered in our evaluation.

4.4.2 Feedback Effect

We first examine the overall retrieval precision of the pseudo feedback models for document-based feedback. The results are summarized in Table 4.2, where the best result for each row is highlighted. As we see, both PRM1 and PRM2 significantly outperform the basic KL-divergence retrieval model in terms of MAP. In addition, PRM1 and PRM2 are also significantly better than RM3 across data sets. For example, the relative improvements of PRM1 over NoFB are 9.0% on Terabyte06 and 13.5% on ClueWeb09 in terms of average precision, which are much larger than the corresponding improvements achieved by RM3 (only 2.8% and 5.9% respectively). RM3 improves Pr@10 over NoFB in neither dataset; however both PRM1 and PRM2 often improve Pr@10, though not significantly. Besides, comparing PRM1 and PRM2, we find that PRM1 is slightly more effective than PRM2.

We are also interested in evaluating if a heuristic passage-based feedback (i.e., RM3-p) can work as well as PRM, since both PRM1 and PRM2 essentially can be regarded as achieving a soft effect of passage feedback. Moreover, we can also use PRM1 and PRM2 for "hard" passage feedback in a way as RM3-p does, which leads to PRM1-p and PRM2-p respectively. So we further compare the average precision of PRM1, PRM2, PRM1-p, PRM2-p, and RM3-p in Table 4.3. From the table, it is clear that PRM1, PRM2, PRM1-p and PRM2-p all outperform RM3-p significantly in most cases, suggesting that our model does not only have sound statistical foundation but also works effectively. In addition, we also observe that RM3-p behaves quite differently in two datasets: it beats RM3 on ClueWeb09 but loses

Collection	Metric	NoFB	RM3	PRM1	PRM2
Terabyte06	MAP	0.3047	0.3131	**0.3322**$^{*+}$	0.3319^{*+}
	Pr@10	0.5367	0.5041	0.5306	**0.5490**$^{+}$
	Pr@30	0.4653	0.4660	**0.4884**$^{+}$	0.4871^{+}
	Pr@100	0.3547	0.3576	0.3671^{*+}	**0.3741**$^{*+}$
ClueWeb09	eMAP	0.0713	0.0755	**0.0809**$^{*+}$	0.0786^{*+}
	eP@10	0.2371	0.2307	**0.2418**$^{+}$	0.2377^{+}
	eP@30	0.2433	0.2486	**0.2536**$^{*+}$	0.2525^{*+}
	eP@100	0.2216	0.2283	**0.2356**$^{*+}$	0.2325^{*+}

Table 4.2: Comparison of different pseudo feedback models for document-based feedback. '*' and '+' mean the corresponding improvements over NoFB and RM3 are significant respectively.

Collection	RM3-p	PRM1	PRM2	PRM1-p	PRM2-p
Terabyte06	0.3077	0.3322*	0.3319*	0.3331*	0.3290*
ClueWeb09	0.0781	0.0809*	0.0786	0.0800*	0.0798*

Table 4.3: MAP/eMAP comparison of passage-based feedback methods. '*' means the corresponding improvement over RM3-p is significant.

to RM3 on Terabyte06. However, all the four variations of PRM perform better than RM3 consistently. Finally, it is also interesting to see that PRM1 and PRM2 work similarly to PRM1-p and PRM2-p respectively, which may mean that PRM1 and PRM2 have already achieved successfully an effect of passage-based feedback by assigning weights to different positions, so it does not bring too much additional benefit to apply PRM to passages explicitly.

Next we examine the robustness to the parameter setting in PRM on the Terabyte06 collection.

4.4.3 Robustness Analysis

In PRM1 and PRM2, there is a parameter σ inherited from the positional language model to control the propagation range, which would influence the effect of term position and term proximity. Specifically, if we increase σ to infinity, the effect of term position and proximity will be disabled. However, if we decrease this parameter to a finite value, term position and proximity will play an important role in PRM. We fix other parameters to their default values as trained on Terabyte05 and focus on understanding how σ affects the retrieval performance of PRM1 and PRM2. From Figure 4.2 (left), we can see that, as long as σ is in the range of [100, 1000], both PRM1 and PRM2 outperform RM3 clearly. Indeed, by setting σ around 200, we can often obtain the optimal performance for both PRM1 and PRM2. This result confirms the observation in previous Chapter 3. In addition, comparing PRM1 and PRM2, PRM2 seems to be less sensitive to σ.

Next, the positional language model is smoothed using Jelinek-Mercer method to estimate PRM. The smoothing is controlled by a parameter λ. When $\lambda = 0$, we are using the pure positional language model, while if $\lambda = 1$, we completely ignore the position and proximity evidence so that every position will receive the same weight. Again, we fix other parameters and show in Figure 4.2 (right) how the average precision changes under different λ. The

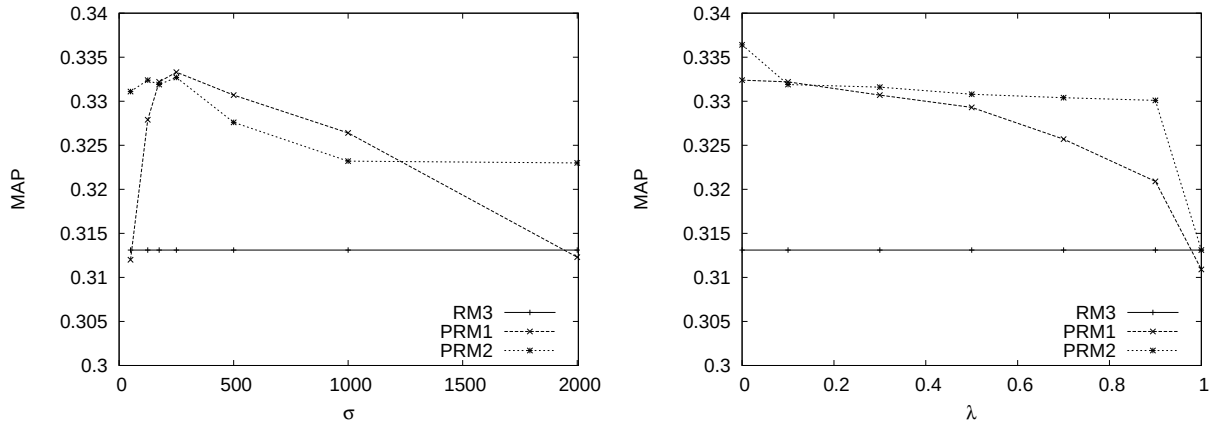

Figure 4.2: Sensitivity to the propagation range σ (left) and the smoothing parameter λ (right) of PRM.

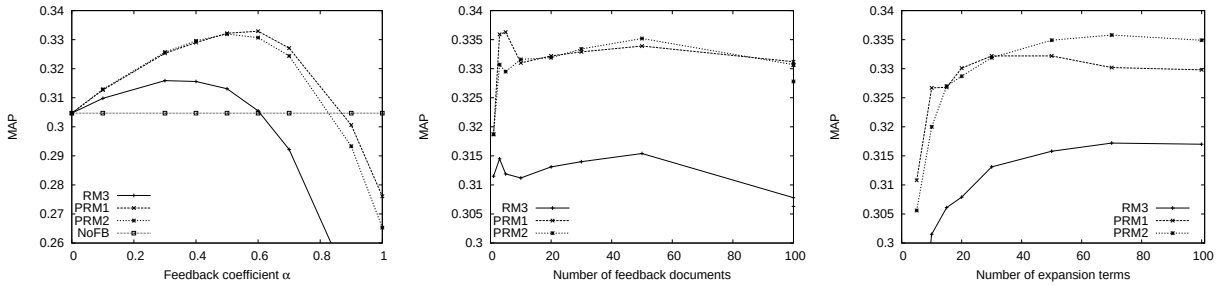

Figure 4.3: Sensitivity to the feedback interpolation coefficient α (left), the number of feedback documents (middle), and the number of expansion terms (right) of different pseudo feedback methods

experiment results indicate that when λ is set to around 0.1, both PRM1 and PRM2 achieve their optimal performance. However, PRM1 and PRM2 always outperform RM3 with $\lambda < 1$. Comparing PRM1 and PRM2, we see again that PRM2 seems to be more robust.

Recall that we interpolate the feedback model with the original query model. The interpolation is controlled by a coefficient α. When $\alpha = 0$, we are only using the original query model (i.e., no feedback), while if $\alpha = 1.0$, we completely ignore the original query model and use only the estimated feedback model. We fix other parameters and show in Figure 4.3 (left) how the average precision changes according to the value of α. We can see that both PRM1 and PRM2 are clearly better than RM3 with different α values. And the optimal α for all the methods seems to be in a range around 0.5. Besides, it is also interesting to observe that the pure feedback model results ($\alpha = 1.0$) of PRM1 and PRM2 are much better than that of RM3, suggesting that the positional relevance model can lead to a more accurate query model. Finally, comparing PRM1 and PRM2, the former seems to be slightly more effective.

We further compare the robustness of different methods w.r.t. the number of feedback documents. We change the number of feedback documents from 1 to 200. The MAP results are shown in Figure 4.3 (middle). We notice that PRM1 and PRM2 are more robust to the number of feedback documents as compared to RM3. It is also interesting to

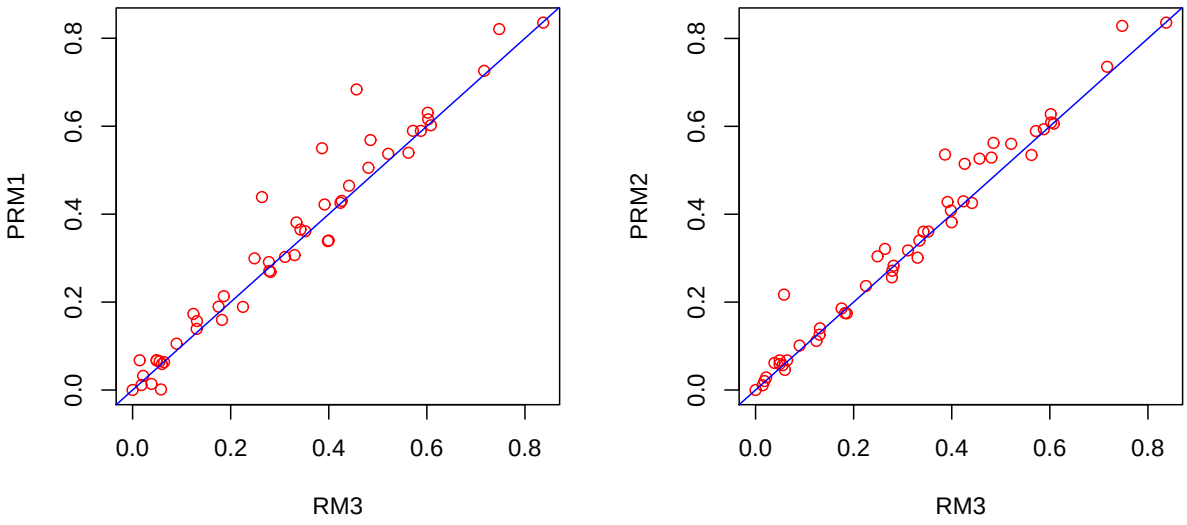

Figure 4.4: MAP Plot of PRM1 (left) and PRM2 (right) as compared to RM3 on Terabyte06

see there is almost no performance decrease of PRM1 and PRM2 even when we set the parameter to 200, suggesting that the proposed positional relevance model works better in tolerating noisy information. Moreover, with only 1 feedback document, PRM1 and PRM2 have already been able to outperform RM3, no matter how many feedback documents RM3 uses, which may indicate that our methods can identify good feedback terms more accurately by assigning position-dependent weights.

Additionally, we also compare the sensitivity of different methods to the number of expansion terms in Figure 4.3 (right). We vary the number of terms from 5 to 100, and observe that both PRM1 and PRM2 can achieve a very effective performance with only 10 expansion terms, while RM3 needs 70 terms, but even so, its performance is still not as good as our methods with 10 terms. This would be another advantage of our methods since fewer expansion terms mean higher efficiency, which is very important for retrieval systems.

To further see the robustness of our methods on individual queries, we plot the MAP of PRM1 versus RM3 and PRM2 versus RM3 on Terabyte06 in Figure 4.4. It is interesting that the proposed methods, particularly PRM2, are quite robust; they improve most of the queries clearly with only a small number of queries decreased slightly.

4.5 Summary

We proposed a novel positional relevance model (PRM) for pseudo-relevance feedback. The PRM exploits term *position* and *proximity* evidence to assign more weights to words closer to query words based on the intuition that words closer to query words are more likely to be consistent with the query topic. Specifically, PRM generalizes the

relevance model to aggregate the associations between a word and query words at the position-level in a probabilistic way. We also developed two methods to estimate the PRM based on different generative models.

Experiment results on two large web data sets show that the proposed PRM is quite effective and robust and performs significantly better than the state of the art relevance model in both document-based feedback and passage-based feedback. Compared to the relevance model, the proposed models are also less sensitive to the setting of various parameters, such as feedback coefficient, number of feedback documents, and number of expansion terms. Comparing the two estimation methods of PRM, the first method (PRM1) appears to be more effective, while the second (PRM2) tends to be more robust. Both methods achieve its optimal retrieval performance when setting the σ value in a range around 200 and λ to around 0.1.

There are many interesting future research directions to explore. One of the most interesting directions is to further study whether setting a term-specific and/or query-specific σ can further improve performance. Another interesting direction is to study how to optimize σ automatically based on the layout of web pages. Improving the estimate of other components in PRM (e.g., the probability of choosing a position in a document) would also be interesting.

Chapter 5

A Boosting Approach to Improving Query Language Models

5.1 Introduction

We have presented a positional relevance model to improve the *effectiveness* of pseudo-relevance feedback approaches for estimating query language models in the previous Chapter. Besides effectiveness, *robustness* of a retrieval model is also very important for any real retrieval systems. In fact, although traditional pseudo-relevance feedback techniques generally improve retrieval performance (e.g., AP) on average, they are not robust in the sense that they tend to help some queries, but hurt other queries [38, 22], limiting its usefulness in real retrieval applications. Thus an important, yet difficult challenge is to *improve the overall effectiveness of pseudo-relevance feedback without sacrificing the performance of individual queries too much*. Although there has been a lot of work on pseudo-relevance feedback, little work has been devoted to address this issue, with only a few exceptions, e.g., [22]. In [22], the authors tried to reduce feedback failures in a constrained optimization approach. However their work was only able to optimize an objective function loosely related to the effectiveness and robustness of pseudo-relevance feedback.

In this chapter, we propose a novel learning algorithm, FeedbackBoost, based on the boosting framework to improve pseudo-relevance feedback through combining a set of basis feedback algorithms optimally using a loss function defined to *directly measure both robustness and effectiveness*, which has not been achieved in any previous work on pseudo-relevance feedback. Specifically, like all other boosting algorithms [35, 92, 34, 115], FeedbackBoost iteratively selects and combines basis feedback methods. In each iteration, a basis feedback method is selected to improve those queries on which the already selected basis feedback methods perform poorly in terms of both effectiveness and robustness. At last, FeedbackBoost uses a linear combination of these basis feedback methods as its final feedback model.

There are several important differences between our work and previous work on improving pseudo-relevance feedback: (1) we cast the pseudo-relevance feedback problem as an optimization problem that can be solved in a supervised way; (2) we propose a novel objective function that directly measures the effectiveness and the number of failure cases of pseudo-relevance feedback; (3) FeedbackBoost can incorporate potentially many different basis feedback methods as features in the model, making it a general optimization framework for pseudo-relevance feedback;

(4) FeedbackBoost does not introduce any extra parameter that needs to be manually tuned.

As an application, we apply FeedbackBoost to improve pseudo-relevance feedback based on language models for estimating query language models through combining different document weighting strategies. One cause of the low robustness and effectiveness in pseudo-relevance feedback is that the feedback documents are simply assumed to be relevant whereas in reality, not all of them are relevant. Thus one way to improve pseudo-relevance feedback would be to assign appropriate weights to these documents. Indeed, our previous work [69] has already shown that with relevance-based document weighting, the relevance model [61] tends to be more robust and effective than alternative models for feedback with language models. In the existing work, however, such weighting is generally based on one heuristic or another, and is not optimized directly to improve feedback. Our main idea is to combine a variety of feedback methods each with a different strategy for document weighting under the framework of FeedbackBoost. Although we only try to leverage feedback methods with different document weighting methods in this work, the proposed boosting framework can potentially accommodate many other basis feedback methods.

We evaluate our method using two representative large test sets and compare FeedbackBoost with multiple baseline methods. The experiment results demonstrate that the proposed FeedbackBoost algorithm can improve average precision significantly and meanwhile reduce the number and magnitude of feedback failures dramatically as compared to two representative pseudo-relevance feedback methods based on language models, the mixture model and the relevance model. We also compare our algorithm with a recently proposed constrained optimization approach to robust feedback, and the results show that our method is more robust. In addition, we compare FeedbackBoost with a traditional learning to rank approach applied for pseudo-relevance feedback and observe that FeedbackBoost works clearly better.

5.2 Related Work

Pseudo-relevance feedback has been extensively studied in the literature, as discussed in Section 4.2. Most existing pseudo-relevance feedback algorithms aim at improving average precision alone but rarely address the robustness issue. In contrast, our work attempts to improve both average precision and robustness at the same time.

A few previous studies also attempted to improve the robustness of pseudo-relevance feedback [103, 100, 22]. Tao and Zhai [103] used a regularized EM algorithm to reduce the parameter sensitivity of the mixture-model feedback but did not minimize the feedback failures. Soskin et al. [100] leveraged multiple relevance models [61] in a heuristic unsupervised way to improve feedback performance. However, their method is not guaranteed to optimize the combination of feedback algorithms. Collins-Thompson [22] also tried to reduce feedback failures in an optimization framework. However this work was only able to optimize an objective function loosely related to the robustness of

pseudo feedback. In contrast, we propose a general machine learning framework to directly optimize both the robustness and effectiveness of pseudo feedback, which can incorporate existing methods, such as [103], [100], and [22], as features. In this sense, our work offers a unified framework that can be used to potentially combine all the existing pseudo feedback methods.

Selective feedback [4] and adaptive feedback [68] are another stream of work to improve the robustness of pseudo feedback, where the idea is to disable query expansion if query expansion is predicted to be detrimental to retrieval [4] or to adaptively set the amount of query expansion in a per-query way [68]. However, these methods are not as general as our proposed framework. Besides, our work and the selective/adaptive feedback method are complementary in the following sense: our work can construct a strong ensemble feedback method, which can be used by selective/adaptive feedback to further improve its performance, while selective/adaptive feedback methods can be incorporated into our framework as features for boosting.

Recently, learning to rank [47, 34, 12, 115, 18, 125, 62] has attracted much attention in IR. Our work can also be regarded a novel use of machine learning to leverage multiple feedback-related features to improve ranking. However, a main difference of our work from traditional work on learning to rank is that we design a novel learning algorithm to directly optimize both robustness and effectiveness of pseudo feedback (novel objective function). Another difference is that most learning to rank work learns optimal ways to combine retrieval functions but fails to improve the query representation. Our work, however, uses machine learning to improve a content-based query representation. Therefore, our study is orthogonal to the existing learning to rank work, and existing learning to rank algorithms can be used to learn a retrieval function on the basis of our improved query representation to further improve retrieval performance.

Machine learning was also introduced to improve pseudo feedback through selecting good expansion terms [17] or good feedback documents [41]. However, neither work attempted to directly optimize robustness of pseudo feedback.

Boosting is a general method for improving the accuracy of supervised learning. The basic idea of boosting is to repeatedly construct "weak learners" by re-weighting training data and form an ensemble of weak learners so that the total performance of the ensemble is "boosted". Freund and Schapire have proposed the first and most popular boosting algorithm called AdaBoost for binary classification [35]. Extensions of boosting have been made to deal with the problems of multi-class classification [92], ranking [34, 115, 125], etc. Our work can be viewed as a novel extension of the boosting framework to improve pseudo-relevance feedback.

5.3 Problem Formulation

Given a query q_i and a document collection C, a retrieval function F returns a ranked list of m documents $\mathbf{d} = \{d_1, \cdots, d_m\}$, where d_j denotes the j-th ranked document in the ranked list. In pseudo feedback, we assume the

top-n documents are "relevant", and construct a feedback model $\phi_t(q_i, \mathbf{d}, n, C)$, or $\phi_t(q_i)$ for short, for query q_i by exploiting those assumed "relevant" documents. Here ϕ_t can be any pseudo feedback method that is able to output an improved query representation, and we call it a *weak* or *basis* feedback method; $\phi_t(q_i)$ is essentially an expanded representation of the original query q_i, and we call it a *weak* or *basis* feedback model for q_i. In general, the format of $\phi_t(q_i)$ would depend on the retrieval function F; for example, in the vector space model, $\phi_t(q_i)$ is represented as a vector of weighted terms, while in language modeling approaches, it is represented as a word distribution. We can use $\phi_t(q_i)$ as the new query and apply F to retrieve another ranked list of documents $\mathbf{d}'$.

Given a performance measure E and the relevance judgments set $J(q_i)$ for query q_i, we can compute the performance scores for the original query q_i and for the expanded query $\phi_t(q_i)$, which will be denoted as $E\left(F(q_i), J(q_i)\right)$ and $E\left(F(\phi_t(q_i)), J(q_i)\right)$, respectively, and will be represented as $E(q_i)$ and $E(\phi_t(q_i))$ in the rest of the chapter for conciseness. We choose the widely accepted average precision (AP) as the performance measure E in this chapter, though the proposed algorithm can in principle also work with other measures.

Although many pseudo feedback methods have been shown to improve the performance of a retrieval system on average, they all share a common deficiency, i.e., the average performance gain always comes inevitably at the cost of (sometimes significantly) degraded performance of some queries. That is, while on average, $\frac{1}{|Q|}\sum_{i=1}^{|Q|} E(\phi(q_i)) > \frac{1}{|Q|}\sum_{i=1}^{|Q|} E(q_i)$, it is almost always the case that for some queries, $E(\phi(q_j)) < E(q_j)$. Indeed, it has been a long-standing difficult challenge to improve the robustness of pseudo feedback so that we can improve average performance without sacrificing the performance of individual queries too much.

In this chapter, we propose to use a learning method to address this problem. Specifically, given a query q_i, we assume there are a variety of basis feedback models $\phi_k(q_i)$ based on different feedback methods $\Phi = \{\phi_1, \cdots, \phi_m\}$, and our main idea is to combine a set of such basis feedback models $\phi_k(q_i)$ into a single feedback model $H(q_i)$ called the *final* or *combined* feedback model to reduce feedback failures:

$$H(q_i) = \sum_{k=1}^{t} \alpha_k \phi_k(q_i) \tag{5.1}$$

A linear combination is chosen because the final feedback model $H(q_i)$ should be in the same format as that of each basis model $\phi_k(q_i)$; that is, if $\phi_k(q_i)$ is a vector of weighted terms, so is $H(q_i)$, while if $\phi_k(q_i)$ is a language model, $H(q_i)$ should also be a language model by normalizing the learned α_k ($k = 1, \cdots, t$) to make them sum to 1.

Our main motivation of combining multiple feedback methods is that different basis feedback methods often have relative strengths for different query topics and thus are complementary to each other. To illustrate it, we examine 46 basis feedback methods constructed using methods described in Section 5.5. The results are presented in Figure 5.1. We can see that many feedback methods indeed have relative strengths on different queries (e.g., topic 708 and

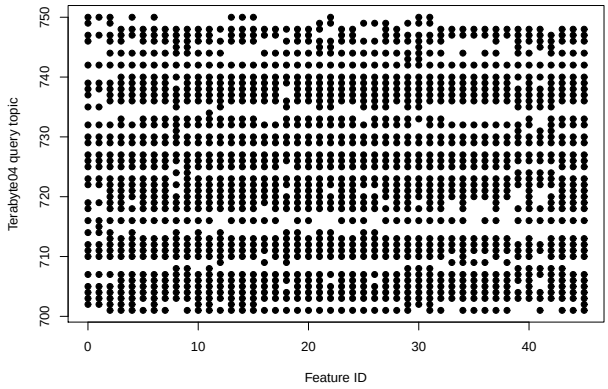

Figure 5.1: Plot of each query w.r.t. different weak feedback methods, where '•' indicates that the corresponding weak feedback improves performance.

709). It seems there are only 3 very hard queries that no feedback method can help, while for other queries, there are always some successful feedback methods. Thus, constructing an ensemble feedback method H would be an effective strategy to reduce feedback loss. For example, suppose we have two weak feedback methods ϕ_1 and ϕ_2; ϕ_1 improves 0.2 and -0.1 (i.e., decreases by 0.1) on q_1 and q_2 respectively, while ϕ_2 improves -0.1 and 0.2 on q_1 and q_2 respectively. So the fbloss scores are 0.5 for both ϕ_1 and ϕ_2 on these two queries. However, an ensemble method $\alpha_1\phi_1 + \alpha_2\phi_2$ would probably have 0 fbloss while still achieving better or comparable average improvement if the combination coefficients α_1 and α_2 are chosen appropriately.

Our goal is to minimize the number of query instances for which the retrieval performance is decreased by the final feedback model $H(q_i)$ as compared to the original query q_i. The learning algorithm that we study attempts to find an H with a small number of query failures, a quality called the *feedback loss* and denoted by $\text{fbloss}_D(H)$. Formally,

$$\text{fbloss}_D(H) = \sum_{i=1}^{|Q|} D(q_i) \cdot I\{E(H(q_i)) < E(q_i)\} \tag{5.2}$$

Here and throughout this chapter, the indicator function $I\{\pi\}$ is defined to be 1 if the predicate π holds and 0 otherwise. $D(q_i)$ is the weight of q_i, and we set it initially to uniform, i.e., $D(q_i) = 1/|Q|$ so that all queries are equally important; later the query weights would be updated iteratively in a boosting framework [35] so that queries that do not perform well would contribute more to the loss function more. We will also show later in Section 5.4.1 that the feedback loss can be bounded by the degradation of retrieval precision, which means that the feedback loss can actually measure both feedback failures and retrieval effectiveness, which is necessary in order to ensure both robustness and effectiveness.

In the following section, we will discuss how to optimize the combination of weak feedback methods so as to minimize fbloss on some training data, formally,

$$\arg\min_{\{\alpha_{1:t}\}} \sum_{i=1}^{|Q|} D(q_i) \cdot I\left\{E(\sum_{k=1}^{t} \alpha_k \cdot \phi_k(q_i)) < E(q_i)\right\} \tag{5.3}$$

5.4 A Boosting Approach to Improving Pseudo-Relevance Feedback

With only a few basis feedback methods, it is possible to optimize their combination through manual parameter tuning. However, manual tuning is infeasible if there are many basis feedback methods as is often the case. Inspired by the AdaBoost [35], we devise a boosting approach, referred to as "FeedbackBoost", to solve this problem.

5.4.1 Minimizing Feedback Loss via Forward Stagewise Additive Modeling

Ideally we want to minimize the feedback loss on the training data as shown in Equation 5.3. However, it is difficult to solve this optimization problem because the combination is inside a retrieval function and not differentiable. To simplify this problem, we make an assumption that the feedback loss of the combined feedback H is less than or equal to that of a linear performance combination of the corresponding basis feedback methods. Formally,

$$\sum_{i=1}^{|Q|} D(q_i) \cdot I\left\{E(\sum_{k=1}^{t} \alpha_k \cdot \phi_k(q_i)) < E(q_i)\right\} \leq \sum_{i=1}^{|Q|} D(q_i) \cdot I\left\{\left[\sum_{k=1}^{t} \frac{\alpha_k}{\sum_{j=1}^{t} \alpha_j} E(\phi_k(q_i))\right] < E(q_i)\right\} \tag{5.4}$$

Although we have not found a theoretical proof for the above inequality, we can empirically guarantee this assumption to be true in our algorithms since we can automatically adjust the algorithms to make sure this assumption holds (as shown in the step 5 of the pseudo code of the algorithm). In practice, this assumption turns out to work well. One possible explanation is that different feedback models often complement to each other, so the performance of a mixture model with reasonable coefficients is often better than the performance of any single model and thus also better than the weighted average performance of these single models. An example is that an appropriate interpolation of a pseudo feedback model and the original query model usually works better than either single model [120, 1, 69]. Thus we will minimize the following upper bound of the feedback loss.

$$\arg\min_{\{\alpha_{1:t}\}} \sum_{i=1}^{|Q|} D(q_i) I\left\{\sum_{k=1}^{t} \alpha_k E(\phi_k(q_i)) < \sum_{k=1}^{t} \alpha_k E(q_i)\right\} \tag{5.5}$$

However the above optimization problem is still hard to handle, because the indicator function I is non-continuous. Fortunately, we know that $I\{x < y\} \leq e^{y-x}$ for all real x and y, so, instead of solving Equation 5.5 directly, we can alternatively minimize its upper bound below, which is essentially a measure of retrieval performance degradation.

$$\arg\min_{\{\alpha_{1:t}\}} \sum_{i=1}^{|Q|} D(q_i) \exp\left(\sum_{k=1}^{t} \alpha_k E(q_i) - \sum_{k=1}^{t} \alpha_k E(\phi_k(q_i))\right) \tag{5.6}$$

Now we can address this problem using forward stagewise additive modeling, which is an effective strategy to find an approximate solution to an optimization problem through sequentially adding new basis method without adjusting

Algorithm 1 The FeedbackBoost algorithm

Require:

Query set $\{q_i, J(q_i)\}_{i=1}^{|Q|}$, retrieval model F, retrieval performance measure E, and the number of iterations T;
A set of basis feedback methods $\Phi = \{\phi_1, \cdots, \phi_m\}$;
Initial distribution D_1 over training queries: $D_1(i) = 1/|Q|$;

Ensure:

1: **for** $t = 1, 2, \cdots, T$ **do**
2: Select basis feedback method ϕ_t with weighted distribution D_t on training queries using Formula 5.14; if there is no ϕ_t selected, break;
3: Compute $\text{Eloss}_t(\phi_t)$ using Formula 5.9.
4: Choose α_t based on Formula 5.12 and $\text{Eloss}_t(\phi_t)$.
5: While the inequality in Formula 5.4 does not hold, goto step 2 and choose the next optimal basis feedback as ϕ_t; if there is no ϕ_t that satisfies the inequality, break;
6: Update D_{t+1} using Formula 5.8;
7: **end for**
8: Output the final feedback: $H = \sum_{t=1}^{T} \alpha_t \phi_t$;

those already selected methods [39]. By forward stagewise additive modeling, the above formula can be expressed as follows where we would iteratively choose α_t and ϕ_t, as well as adjust $D_t(q_i)$:

$$\arg\min_{\{\alpha_t, \phi_t\}} \sum_{i=1}^{|Q|} D_t(q_i) \cdot \exp\left(\alpha_t \cdot [E(q_i) - E(\phi_t(q_i))]\right) \tag{5.7}$$

where

$$D_t(q_i) \propto \frac{D_{t-1}(q_i) \cdot \exp\left(\alpha_{t-1} \cdot [E(q_i) - E(\phi_{t-1}(q_i))]\right)}{Z_t} \tag{5.8}$$

Here, Z_t is a normalization factor to make D_t a distribution; such a normalization operation does not affect the choosing of α_t and ϕ_t, since D_t depends neither on α_t nor ϕ_t in the forward stagewise additive modeling [39]. D_t is defined in a recursive way, where $D_1(q_i) = D(q_i) = 1/|Q|$ is the initial weight for query q_i. After T iterations, we can obtain the desired combined feedback model as $H(q_i) = \sum_{t=1}^{T} \alpha_t \phi_t$.

5.4.2 FeedbackBoost

FeedbackBoost is designed to find a solution to the optimization problem in Formula 5.7 using a boosting approach. Specifically, like all other boosting algorithms, FeedbackBoost operates in rounds. At each round, we calculate a distribution of weights over training queries. In fact, D_t can be regarded as a weight that is applied to each query after $t-1$ iterations. From Equation 5.8, we can see that D_t will put more weights on queries that are hurt more by previously selected basis feedback methods.

We next select a basis feedback method ϕ_t that works well on those highly-weighted queries, and the selection of

this basis feedback method is to optimize an objective function that is defined to directly measure the feedback loss. Specifically, we define the weighted performance degradation of ϕ at iteration t as

$$\text{Eloss}_t(\phi) = \sum_{i=1}^{|Q|} D_t(q_i) \cdot (E(q_i) - E(\phi(q_i))) \tag{5.9}$$

And the feedback method ϕ_t with the minimum Eloss_t will be selected. Theoretical analysis is provided as follows:

The performance measure E, e.g., AP in this paper, is usually within range $[0, 1]$. Even a measure is beyond this range, we can still normalize it to $[0, 1]$. Therefore, for any basis feedback method ϕ_t, we have the performance degradation $E(q_i) - E(\phi_t(q_i)) \in [-1, 1]$. By the convexity of $e^{\alpha x}$ as a function of x when $x \in [-1, 1]$ [34], we thus have

$$\exp\left(\alpha_t[E(q_i) - E(\phi_t(q_i))]\right) \leq \left(\frac{1 + E(q_i) - E(\phi_t(q_i))}{2}\right)\exp(\alpha_t) + \left(\frac{1 - E(q_i) + E(\phi_t(q_i))}{2}\right)\exp(-\alpha_t) \tag{5.10}$$

So Equation 5.7 can be approximated by

$$\arg\min_{\{\alpha_t, \phi_t\}} \frac{1 + \text{Eloss}_t(\phi_t)}{2}\exp(\alpha_t) + \frac{1 - \text{Eloss}_t(\phi_t)}{2}\exp(-\alpha_t) \tag{5.11}$$

We can easily minimize this equation by setting

$$\alpha_t^* = \frac{1}{2}\log\frac{1 - \text{Eloss}_t(\phi_t)}{1 + \text{Eloss}_t(\phi_t)} \tag{5.12}$$

which indeed makes sense since it suggests that a ϕ_t with less performance degradation will receive a larger weight α_t. Then, by plugging Equation 5.12 into 5.11, we obtain the following optimal basis feedback.

$$\phi_t^* = \arg\min_{\{\phi_t\}} \sqrt{(1 - \text{Eloss}_t(\phi_t))(1 + \text{Eloss}_t(\phi_t))} \tag{5.13}$$

We can see that Equation 5.13 is minimized when $\text{Eloss}_t(\phi_t)$ is close to 1 or -1. With respect to the former case, we would choose a ϕ_t with the largest weighted performance degradation, and α_t will be negative, which, however, does not make sense: if a pseudo feedback ϕ_t leads to large performance degradation, a negative α_t may not make ϕ_t work well. Therefore, $\text{Eloss}_t(\phi_t)$ should be negative to keep the coefficient α_t positive. So we choose a basis pseudo feedback ϕ_t with the smallest negative Eloss_t so far.

$$\phi_t^* = \arg\min_{\{\phi_t\}} \{\text{Eloss}_t(\phi_t)\} \quad \text{subject to } \text{Eloss}_t(\phi_t) < 0 \tag{5.14}$$

5.4.3 Algorithm Summary

To summarize, FeedbackBoost works as follows. The input to the algorithm includes a set of queries and relevance judgments $\{q_i, J(q_i)\}_{i=1}^{|Q|}$, a set of basis feedback methods $\Phi = \{\phi_1, \cdots, \phi_m\}$, a document collection C, a retrieval function F, a retrieval performance measure E, and the iteration number T. FeedbackBoost works in an iterative way: during each iteration t, a basis feedback method ϕ_t is chosen based on its performance on training data with weight distribution D_t, i.e., $\text{Eloss}_t(\phi_t)$. Also the coefficient α_t of ϕ_t is calculated based on $\text{Eloss}_t(\phi_t)$. After that, the query weight distribution D_t is updated by increasing weights on queries for which ϕ_t performs poorly, leading to a new distribution D_{t+1}; D_{t+1} will be used in the next iteration to select ϕ_{t+1} and α_{t+1}. At last, the final feedback model H is created by linearly combining all the selected basis feedback methods. A sketch of the algorithm flow is shown in Algorithm 1.

5.5 Application of FeedbackBoost to Language Models

In order to apply FeedbackBoost, the main task is to design appropriate basis feedback methods $\{\phi\}$. A basis feedback method ϕ_k generally consists of three components: a weighting function h_k to assign weights to feedback documents, a weighting function g_k to calculate the importance of different expansion terms, and the retrieval model F to decide the representation format of the feedback model. Formally, $\phi_k = f(h_k, g_k, F)$. Given a retrieval model F, one feedback method differs from others often because it uses different document and/or term weighting functions [69]. We can thus naturally construct many basis feedback methods by varying the document and/or term weighting functions.

5.5.1 Basis Pseudo-Relevance Feedback Methods based on Language Models

As a specific application, here we discuss how we can apply FeedbackBoost to improve pseudo feedback under the language modeling framework through combining different document weighting strategies. This application is especially interesting because (1) language models deliver state of the art retrieval performance [81, 58]; (2) feedback document weighting has been shown to be a critical factor affecting robustness and effectiveness of pseudo feedback methods [69]. However, our methodology could be applicable to other retrieval models and to optimizing term weighting methods as well, which we leave as future work.

We use the KL-divergence retrieval method [58] as our retrieval model (i.e., F), which scores a document d with respect to a query q by computing the negative KL divergence between the query and the document language model:

$$S(q,d) = -\sum_{w \in q} P(w|q) \log \frac{P(w|q)}{P(w|d)} \tag{5.15}$$

Two important instantiations of basis pseudo feedback methods in language models are the relevance model [61] and the mixture model [120], which are among the most effective and robust feedback techniques based on language models [69].

Relevance Model

The relevance model ϕ_r essentially uses the query likelihood as the weight for document d and takes an average of the probability of word w given by each document language model. Formally, let Θ represent the set of smoothed document models in the pseudo feedback collection $\Omega = \{d_1, \cdots, d_n\}$. The formula of the relevance model is:

$$P(w|\phi_r(q)) \propto \sum_{\theta_d \in \Theta} P(w|\theta_d) \prod_{w' \in q} P(w'|\theta_d) \tag{5.16}$$

Let's denote the original query model as $P(w|q)$. The relevance model $P(w|\phi_r(q))$ can be interpolated with the original query model $P(w|q)$ to improve performance using a interpolation coefficient α. In this paper, we will use the following interpolated model $P(w|\theta_q)$ as the new query model, which is often called RM3 [1]:

$$P(w|\theta_q) = (1-\alpha)P(w|q) + \alpha P(w|\phi_r(q)) \tag{5.17}$$

In Equation 5.16, $h_r(d) = \prod_{w' \in q} P(w'|\theta_d)$ is the query likelihood score of document d, serving for document weighting, and $g_r(w,d) = P(w|\theta_d)$ works for term weighting. If we instantiate the document weighting strategy in a different way, e.g., h'_r, whereas the term weighting strategy is fixed to g_r, it will lead to a different "relevance model" ϕ'_r. Formally

$$P(w|\phi'_r(q)) \propto \sum_{\theta_d \in \Theta} P(w|\theta_d) \cdot h'_r(d) \tag{5.18}$$

which can also be used for feedback after a similar interpolation. Following this way, we can construct a set of relevance-model style basis feedback methods by varying their document weighting strategies.

Mixture Model

In the simple two-component mixture model (SMM) ϕ_m, the words in Ω are assumed to be drawn from two models: (1) background model $P(w|C)$ and (2) topic model $P(w|\phi_m(q))$. Thus the log-likelihood for the entire set of feedback documents is:

$$\log P(\Omega|\phi_m(q)) = \sum_{w \in V} c(w,\Omega) \log((1-\lambda)P(w|\phi_m(q)) + \lambda P(w|C)) \tag{5.19}$$

where V is the word vocabulary, $c(w, \Omega)$ is the count of word w in feedback document set Ω, and $\lambda \in [0, 1]$ is the mixture parameter. The estimate of $\phi_m(q)$ can be computed using the EM algorithm to maximize the log-likelihood. Finally, $\phi_m(q)$ is also interpolated with the original query model $P(w|q)$ to update the query model with a coefficient α. We notice in Formula 5.19 that

$$c(w, \Omega) = \sum_{d \in \Omega} |d| \cdot P(w|d) \tag{5.20}$$

It means that the document length $|d|$ is used as a weight of document d (i.e., $h_m(d) = |d|$) to sum over the term evidence from each feedback document. As in the case of the relevance model, we can also use any other document weighting method h'_m to replace h_m while keeping g_m the same, which will lead to a new family of mixture-model style basis feedback method ϕ'_m.

5.5.2 Document Weighting Strategies

We next introduce a set of document weighting strategies $\{h_t\}$. As we have discussed, each of them can be plugged into the relevance model (Section 5.5.1) or the mixture model (Section 5.5.1), leading to a new basis feedback method ϕ_t.

Relevance Score: Relevance score is shown to be a critical factor for feedback document weighting in a recent work [69]. We explore the use of query likelihood [81] h_1 and BM25 score [87] h_2 for document weighting: w.r.t. the former, the Dirichlet smoothing method [121] with $\mu = 1,000$ is used to smooth the document language model; w.r.t. the latter, we fix $k_1 = 1.2$, $b = 0.5$, and $k3 = 1,000$.

Document Novelty: We estimate a novelty score for each document to reward novel information. Three different methods are proposed: (1) The distance between the centroid of all feedback documents $h_3(d_i) = 1 - \text{cosim}(\vec{d_i}, \frac{1}{k}\sum_{j=1}^{k} \vec{d_j})$, where 'cosim' stands for cosine similarity; (2) The distance between the centroid of all feedback documents ranked before the document $h_4(d_i) = 1 - \text{cosim}(\vec{d_i}, \frac{1}{i-1}\sum_{j=1}^{i-1} \vec{d_j})$. (3) The distance between the most similar document ranked before the document: $h_5(d_i) = 1 - \max_{j=1}^{i-1}\{\text{cosim}(\vec{d_i}, \vec{d_j})\}$.

Query Term Proximity: Query term proximity has been largely ignored in traditional retrieval models [87, 81]. We use the proposed positional language model in Chapter 3 and the minimum pair distance [104] to capture term proximity for improving document weighting: (1) we compute a positional query likelihood score based on the "best-matching" position, i.e., $h_6(d) = \max_{j=1}^{|d_i|} \prod_{w \in q} P(w|d, j)^{c(w,q)}$, where $c(w, q)$ is the count of w in q; (2) we use the normalized minimum pair-wise distance proposed in [104] by setting $\alpha = 1$ which prevents negative document weights, i.e., $h_7(d) = \log(\alpha + \exp(-\delta(q, d)))$.

Document Length: Though the heuristic of document length normalization has been incorporated into the rele-

vance scores [87, 121], we still list it here because it was explicitly used in some existing pseudo feedback methods [69]: (1) raw document length: $h_8(d) = |d|$; (2) reciprocal of the raw document length: $h_9(d) = 1/h_8(d)$; (3) Dirichlet document length: $h_{10}(d) = |d|/(|d| + \mu)$, where we set $\mu = 1,000$; (4) reciprocal of the Dirichlet document length: $h_{11}(d) = 1/h_{10}(d)$.

Besides the above basic document weighting methods, for each h_t, we also introduce some of their variations as our document weighting, including $\exp(h_t)$, $(h_t)^2$, and $\sqrt{h_t}$. Also we include $\log(h_2)$ and $\log(h_8)$, since the values of h_2 and h_8 are usually larger than 1.0 for top-ranked documents. Overall, there are 46 methods in total, all of which are normalized to sum to 1.0 for each query.

5.5.3 Implementation Details

We next apply the FeedbackBoost algorithm to learn an ensemble feedback method. We denote $E_d(\phi(q))$ as the score of document d with respect to a basis feedback method ϕ on query q. In fact, with the KL-divergence retrieval method (Equation 5.15), we only need to retrieve and score documents once for each basis feedback method. Then during the training process, if we need to score a document d using any combined feedback $H' = \sum_{k=1}^{t} \alpha_k \phi_k$, we can do it efficiently by linearly combining the scores of the corresponding basis feedback methods.

$$\begin{aligned} E_d(H'(q)) &\propto \sum_{w \in q} P\left(w | \sum_{k}^{t} \frac{\alpha_k}{\sum_j^t \alpha_j} \phi_k(q)\right) \log P(w|d) \\ &= \sum_{w \in q} \left[\sum_{k=1}^{t} \frac{\alpha_k}{\sum_{j=1}^t \alpha_j} P\left(w|\phi_k(q)\right)\right] \log P(w|d) \\ &\propto \sum_{k=1}^{t} \alpha_k \sum_{w \in q} P(w|\phi_k(q)) \log P(w|d) \\ &\propto \sum_{k=1}^{t} \alpha_k E_d(\phi_k(q)) \end{aligned} \tag{5.21}$$

So training FeedbackBoost would be as efficient as training general AdaBoost algorithm [35].

Besides, during the testing phase, $H(q)$ can also be estimated efficiently. For example, for the relevance model, we can plug the ensemble document weighting method used in $H(q)$ into Formula 5.18 to replace $h'_r(d)$ and estimate $H(q)$ directly; for the mixture model style $H(q)$, we can also replace $|d|$ with the combined document weighting methods in Formula 5.20, which does not affect feedback performance in the experiments. Hence, the efficiency of $H(q)$ for pseudo feedback would be comparable to that of basis feedback algorithms, which is also confirmed empirically.

Collection	Description	#Docs	Training Topics	Validate Topics	Test Topics
Robust04	TREC disk 4&5 (minus CR)	528,155	301-450	601-650	651-700
Terabyte	2004 crawl of .gov domain	25,205,179	701-750	751-800	801-850

Table 5.1: Overview of TREC collections and topics

5.6 Experiments

5.6.1 Experimental Setup

We evaluate our method using the Terabyte Web dataset and a large news dataset Robust04. Only title portions of the topics are taken as queries. For each dataset, we split the available topics into training, validate and test sets, where the training set is used solely for training the algorithms, the validate set is used for tuning the number of iterations T, and the test set is used for evaluation purposes. Table 5.1 shows some document set statistics. The preprocessing of the collections includes stemming using the Porter algorithm and stopword removal using a standard InQuery stoplist.

We train two FeedbackBoost models: in the first one, each basis feedback method implements a different document weighting strategy proposed in Section 5.5.2, while all of them use the same mixture-model style term weighting, and thus we refer to it as **BoostMM**; in the second one, each basis feedback method also implements a different document weighting strategy but all of them share the relevance-model style term weighting, thus the name **BoostRM**. That is, we parameterize Φ in different ways for BoostRM and BoostMM. This design allows us to meaningfully compare BoostMM and BoostRM with the corresponding two baseline feedback methods (i.e., SMM and RM3) and the basic query language model without feedback (labeled as "NoFB"). This set of experiments are mainly to compare FeedbackBoost with traditional pseudo feedback methods. A main hypothesis we would like to test is whether BoostMM and BoostRM are indeed more robust than the corresponding baseline SMM and RM3.

Furthermore, we also compare FeedbackBoost with two other lines of baseline methods. In the first line, we compare FeedbackBoost with another strong baseline representing a recent work on improving robustness of pseudo feedback, i.e., the **REXP-FB** method [22]. In the second line, we are interested in knowing if an existing learning to rank approach can also improve both robustness and effectiveness as much as FeedbackBoost does. So we introduce yet another baseline **AdaRank** [115], which performed well [62]. AdaRank attempts to learn a ranking function through directly optimizing retrieval measures. One variation of AdaRank used in our comparison is **AdaRank.MAP**, which tries to optimize MAP. Since it is non-trivial to directly optimize the proposed fbloss using AdaRank, we further extend AdaRank to optimize a loss function that is similar to fbloss: a novel robustness-related measure E' that measures the performance degradation in feedback: $E'(q_i) = E\left(\sum_{k=1}^{t} \alpha_k \cdot \phi_k(q_i)\right) - E(q_i)$. According to the Formula 6 in [115], AdaRank turns out to minimize an exponential loss function $\sum_{i=1}^{|Q|} \exp\{-E'(q_i)\}$, leading to a new run **AdaRank.FB**. Note that AdaRank.FB is no longer the traditional AdaRank algorithm because the loss function is novel, thus it is a very strong baseline.

We use the Dirichlet smoothing method [121] for all document language models, where we set the $\mu = 1,000$. Besides, as suggested in our previous study [69], we set the mixture noise parameter λ to 0.9 for SMM and all the mixture-model style basis feedback methods. We also set the number of expansion terms to 40. These parameter settings work well and are used in our experiments unless otherwise stated.

It does not make sense to talk about fbloss alone, since we can always get 0 loss for any feedback method by setting the feedback coefficient α to 0. In the traditional evaluation strategy [120, 69], people often tune α to optimize one retrieval precision measure. We thus follow such a strategy to first optimize α in terms of MAP, on the basis of which we then try to reduce fbloss. Specifically, we give a priority to SMM and RM3 to optimize their feedback coefficient α on the training, validate and test sets respectively. However, for all the mixture-model style basis feedback methods, we simply set the feedback coefficients to those optimized for SMM, while for all the relevance-model style basis feedback methods, we use RM3's setting directly.

We are interested in both effectiveness and robustness of pseudo feedback methods, so besides MAP (on top-ranked $1,000$ documents) and Pr@20, we also compare all runs in terms of two robustness measures, the robustness index (RI) and the accumulative loss of retrieval performance (APloss). $\text{RI} = 1 - 2 \cdot \text{fbloss}$, is essentially a transformation of the fbloss proposed in our paper; we show RI instead of fbloss mainly because RI was often used in previous studies, e.g., [22]. APloss is to measure the feedback loss in a finer degree, which is the accumulative AP degradation in failure cases (since AP is used as E in our paper), defined as: $\text{APloss} = \sum_{i=1}^{|Q|} [E(q_i) - E(H(q_i))] \cdot I\{E(H(q_i)) < E(q_i)\}$

5.6.2 Performance of FeedbackBoost

Table 5.2 compares MAP, Pr@20, RI, and APloss for NoFB, SMM, RMM, BoostMM, and BoostRM on the validate and test sets. Note that each time all runs use the same set of feedback documents to make the comparison fair; that is, we fix the base ranking for all runs. Besides, the iteration number T in FeedbackBoost is chosen to minimize the corresponding fbloss on the validate sets. We also vary the number of feedback documents from 20 to 50.

For all cases, we can see that BoostMM and BoostRM significantly improve the robustness over SMM and RM3 respectively. For example, BoostMM *reduces* APloss as compared with SMM by amounts ranging from 50.9% to 77.8% when using 20 feedback documents and from 65.1% to 79.5% when using 50 feedback documents. Moreover, BoostMM and BoostRM also significantly improve MAP over SMM and RM3 in almost all cases. The results demonstrate that the FeedbackBoost algorithm does a good job to improve robustness while still achieving better effectiveness. Moreover, FeedbackBoost works consistently well when we use different number of feedback documents. The performance of FeedbackBoost shows that it is effective to combine multiple feedback methods using our FeedbackBoost to improve both robustness and effectiveness of pseudo feedback.

We next compare FeedbackBoost with AdaRank.MAP and AdaRank.FB. These three algorithms are all trained on

BoostMM Versus SMM

Collection		Metric	NoFB	fbDocCount = 20		fbDocCount = 50	
				SMM	BoostMM	SMM	BoostMM
Validate	Robust04	MAP	0.2850	0.3215*	**0.3447**$^{*+}$	0.2973	**0.3468**$^{*+}$
		Pr@20	0.3310	0.3610	**0.3790**	**0.3850**	**0.3850**
		RI	n/a	0.3600	**0.6400** (+77.8%)	0.0400	**0.6000** (+1400%)
		APloss	n/a	0.6191	**0.2877** (−53.5%)	1.3730	**0.2815** (−79.5%)
	Terabyte	MAP	0.3076	0.3477*	**0.3701**$^{*+}$	0.3445*	**0.3669**$^{*+}$
		Pr@20	0.5410	0.5570	**0.5970**	0.5540	**0.5820**
		RI	n/a	0.4400	**0.7200** (+63.6%)	0.3200	**0.6000** (+87.5%)
		APloss	n/a	0.6036	**0.1667** (−72.4%)	0.6963	**0.2433** (−65.1%)
Test	Robust04	MAP	0.2930	0.3265*	**0.3511**$^{*+}$	0.3067	**0.3453**$^{*+}$
		Pr@20	0.3796	0.4041	**0.4143**	0.3867	**0.4082**
		RI	n/a	0.3878	**0.5102** (+31.6%)	0.2653	**0.4286** (+61.6%)
		APloss	n/a	0.7108	**0.3493** (−50.9%)	1.3429	**0.4231** (−68.5%)
	Terabyte	MAP	0.3012	0.3061	**0.3331**$^{*+}$	0.3067	**0.3298**$^{*+}$
		Pr@20	0.4878	0.4765	**0.5255**	0.4663	**0.5112**
		RI	n/a	0.1429	**0.5102** (+257%)	0.1429	**0.5102** (+257%)
		APloss	n/a	0.6438	**0.1499** (−76.7%)	0.7546	**0.2511** (−66.7%)

BoostRM Versus RM3

Collection		Metric	NoFB	fbDocCount = 20		fbDocCount = 50	
				RM3	BoostRM	RM3	BoostRM
Validate	Robust04	MAP	0.2850	0.3431*	**0.3450***	0.3430*	**0.3490**$^{*+}$
		Pr@20	0.3310	**0.3840**	0.3800	0.3800	**0.3880**
		RI	n/a	0.2800	**0.4800** (+71.4%)	0.3600	**0.5200** (+44.4%)
		APloss	n/a	0.7084	**0.5056** (−28.6%)	0.6421	**0.4155** (−35.3%)
	Terabyte	MAP	0.3076	0.3471*	**0.3661**$^{*+}$	0.3466*	**0.3628**$^{*+}$
		Pr@20	0.5410	0.5840	**0.5890**	**0.5770**	0.5740
		RI	n/a	0.5200	**0.6800** (+30.8%)	0.4800	**0.5600** (+16.7%)
		APloss	n/a	0.9042	**0.2946** (−67.4%)	0.9458	**0.4247** (−55.1%)
Test	Robust04	MAP	0.2930	0.3483*	**0.3537**$^{*+}$	0.3462*	**0.3488**$^{*+}$
		Pr@20	0.3796	0.4010	**0.4122**	**0.4082**	**0.4082**
		RI	n/a	0.3061	**0.4286** (+40.0%)	0.3061	**0.4694** (+53.3%)
		APloss	n/a	0.5736	**0.3662** (−36.2%)	0.5184	**0.4261** (−17.8%)
	Terabyte	MAP	0.3012	0.3187	**0.3287***	0.3188	**0.3311**$^{*+}$
		Pr@20	0.4878	0.4939	**0.5010**	0.4980	**0.5173**
		RI	n/a	0.2245	**0.3469** (+54.5%)	0.1837	**0.3469** (+88.8%)
		APloss	n/a	0.6847	**0.2797** (−59.1%)	0.6608	**0.2614** (−60.4%)

Table 5.2: Comparison of BoostMM and BoostRM with SMM and RM3 respectively. Note that for APloss, lower is better (negative change is good), while for RI, higher is better. '*' and '+' mean the MAP improvement is statistically significant over NoFB and the corresponding baseline feedback method respectively. The improvement of APloss and RI is shown for BoostMM/BoostRM relative to SMM/RM3.

Metric	fbDocCount = 20			fbDocCount = 50		
	AdaRank.MAP	AdaRank.FB	FeedbackBoost	AdaRank.MAP	AdaRank.FB	FeedbackBoost
MAP	0.3451	0.3535	**0.3537**	0.3418	0.3485	**0.3488**
RI	0.3061	0.3878	**0.4286**	0.3061	0.3469	**0.4694**
APloss	0.9776	0.4333	**0.3662**	1.0033	0.6329	**0.4261**

Table 5.3: Comparison of FeedbackBoost, AdaRank.MAP, and AdaRank.FB on the test set of Robust04. Note that AdaRank.FB is different from the traditional AdaRank algorithm, since it is armed with a novel robustness-related loss function.

Collection		NoFB-1	REXP-FB	NoFB-2	BoostRM	BoostMM
Robust04	MAP	0.2152	0.2451	0.2502	0.2617^{+}	0.2752^{+}
	RI	n/a	0.3773	n/a	0.5100	**0.5221**
Terabyte	MAP	0.2736	0.3004	0.2901	0.3086^{+}	0.3230^{+}
	RI	n/a	0.2624	n/a	0.5135	**0.5270**

Table 5.4: Comparison of FeedbackBoost and REXP-FB on the same collections, where cross-validation is used for training. '+' indicates that the improvement of MAP over NoFB-2 is statistically significant.

the same set of relevance-model style basis feedback methods. The iteration number for all the algorithms are chosen to minimize fbloss on the validate set. We present the experiment results in Table 5.3. One interesting observation is that AdaRank.FB is more effective than AdaRank.MAP, suggesting that the proposed robustness-related measure is better than MAP as an objective function to improve pseudo feedback: one possible explanation is that the average precision does not work well to indicate the room for improvement of a query, so focusing on queries with lower average precision may not be a good strategy to fully exploit the potential of different queries; however, our new measure would be able to capture more precisely the potential room of a query through comparing its performance with the baseline performance. Moreover, we also see that FeedbackBoost is clearly better than AdaRank.FB, though they use similar objective functions, sugggesting that our optimization framework is more effective for pseudo feedback.

We finally compare FeedbackBoost with a state-of-the-art feedback method, REXP-FB [22], which attempted to reduce failures of pseudo feedback but was only able to optimize an indirect objective function. They also used Terabyte and Robust04 as their test collections. So we used a 3-fold (701-750, 751-800, and 801-850) and a 2-fold (301-450 and 601-700) cross validation method to evaluate FeedbackBoost on the whole Terabyte and Robust04 topics respectively in order to compare with their reported numbers. In this comparison, we use the same collection and parameter settings as were used in [22]. The comparison is reported in Table 5.4.

There are clear performance improvements of our baseline runs (NoFB-2) over theirs (NoFB-1), probably because we use a latest version of Indri search engine (2.10) [1]. We still can see that, in terms of relative improvements, BoostMM performs similarly to REXP-FB, although REXP-FB used a lot of constraints to improve the selection of expansion terms while we only use the default term weighting. However, the most interesting observation is from the comparison of RI, which may be more comparable across systems. We can see that our algorithms achieve a significantly *higher RI* in all cases, even though our baseline run is even harder to beat. This finding confirms the conclusion in [69] that document weighting plays a key role in affecting the robustness of feedback. Furthermore, it suggests that our way of directly optimizing robustness indeed works more effectively.

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

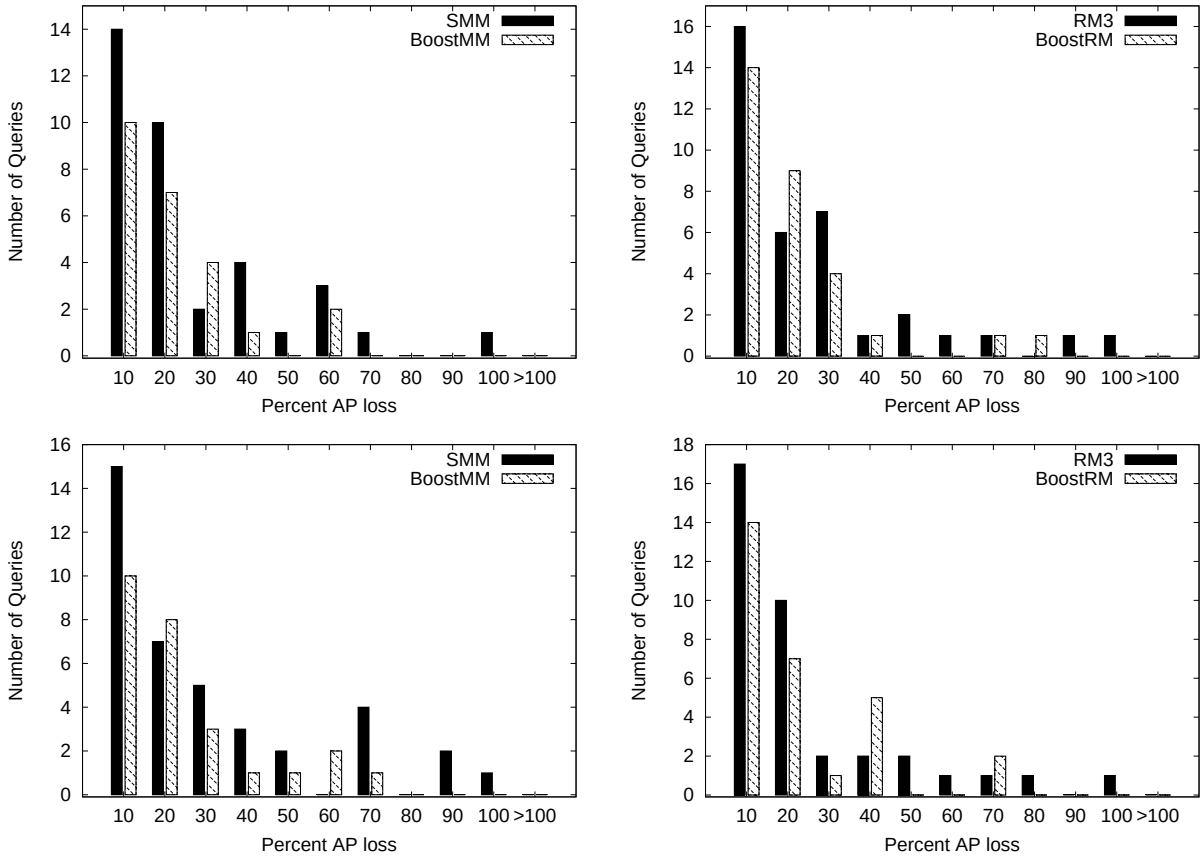

Figure 5.2: Histograms that compare robustness of SMM vs BoostMM (first and third) and RM3 vs BoostRM (second and fourth) on the combination of two test sets. 20 (50) feedback documents are used in the left (right) two histograms respectively.

5.6.3 Robustness Histograms

To examine in details how badly queries are hurt by a pseudo feedback algorithm, we show in Figure 5.2 the robustness histograms by combining two test sets (query 651-701 and query 801-850). The x-axis represents the individual APloss in percentage (i.e., $[E(q) - E(\phi(q))]/E(q)$, where ϕ is the corresponding feedback method) for individual failure queries, and the y-axis stands for the number of queries with the corresponding percentage APloss. We can see that for the worst cases, where a query's AP is decreased by more than 40%, both BoostMM and BoostRM perform much better than SMM and RM3 respectively. It suggests that the proposed FeedbackBoost algorithm indeed concentrates more on difficult queries that are hurt seriously by baseline feedback methods, i.e., SMM and RM3, and the significant reduction of APloss and improvement of RI shown in Table 5.2 could be mainly due to the elimination of those worst cases.

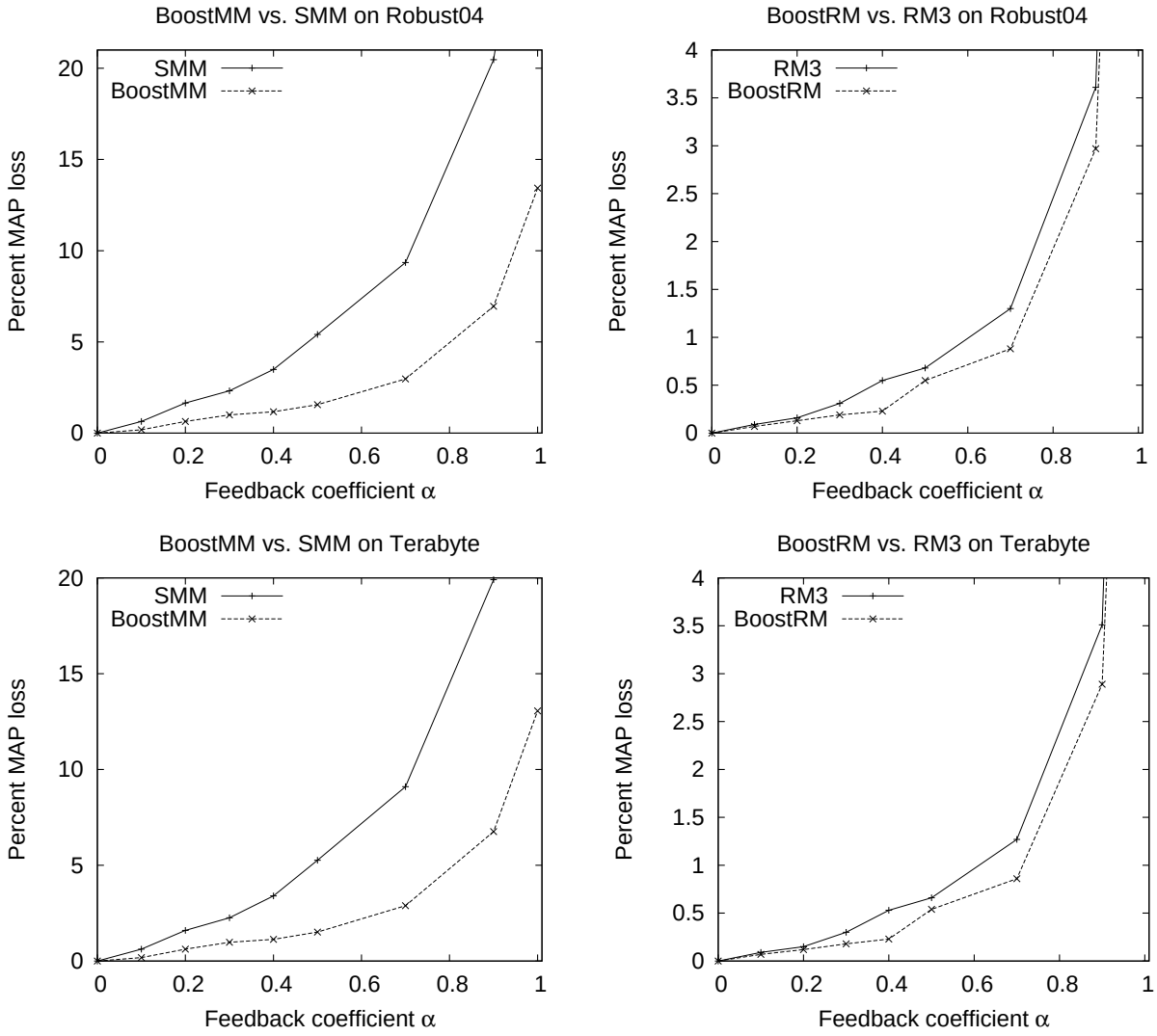

Figure 5.3: Sensitivity of the percentage degradation of MAP in failure queries to the feedback coefficient α. SMM versus BoostMM and RM3 versus BoostRM are shown in the odd and even-number figures respectively.

5.6.4 Parameter Sensitivity

Usually there is a tradeoff between robustness and effectiveness: if we use a smaller feedback coefficient α, there would be fewer feedback failures, but we may not fully improve the effectiveness; if we increase this α to some appropriate value, the overall retrieval precision could be optimized, which, however, may lead to more feedback failures. Although we have shown that FeedbackBoost improves both robustness and effectiveness at the same time, we are still interested in how the performance of FeedbackBoost interacts with α (where α in FeedbackBoost is used to control the feedback interpolation of each basis feedback method involved.) We thus draw the sensitivity curves for FeedbackBoost in terms of the percentage MAP degradation and the overall MAP improvement in Figure 5.3 and 5.4 respectively. The percentage MAP degradation is essentially the relative APloss, i.e., APloss$/[\sum_i^{|Q|} E(q_i)]$; the overall MAP improvement of ϕ is defined as $[\sum_i^{|Q|} E(\phi(q_i))]/[\sum_i^{|Q|} E(q_i)] - 1.0$. We use 50 feedback documents in all curves. The curves clearly show that FeedbackBoost consistently improves the robustness and effectiveness over two baseline algorithms. Additionally, our algorithm is also less sensitive to α in both curves, and setting α around 0.8 often leads to a large improvement in precision with only a small amount of failures.

Finally, we show in Figure 5.5 the curves of performance changes as we increase the iteration number T. We see that the APloss decreases steadily and quickly as the training goes on, until it reaches its plateau. In our experiments, we can usually find the best parameter T within 100 rounds.

5.7 Summary

In this paper, we propose a novel learning algorithm, FeedbackBoost, based on the boosting framework to improve pseudo feedback. A major contribution of our work is to optimize pseudo feedback based on a novel loss function that *directly measures both robustness and effectiveness*, which has not been achieved in any previous work.

The experiment results show that the proposed FeedbackBoost algorithm can improve average precision effectively and meanwhile reduce the number and magnitude of feedback failures dramatically as compared to two representative pseudo feedback methods based on language models, the mixture model and the relevance model. We also compare our algorithm with a recently proposed robust feedback method, and the results show that our method is more robust. In addition, we compare FeedbackBoost with a well-performing learning to rank approach applied for pseudo feedback and observe that FeedbackBoost works clearly better. These results show that the proposed FeedbackBoost is more effective and robust than any of the existing method for pseudo feedback, including both traditional pseudo feedback methods and new learning-based approaches.

Our work can be extended in several ways. First, in our current work, we only use basis feedback methods with different document weighting strategies, so a straightforward future work is to also introduce term weighting

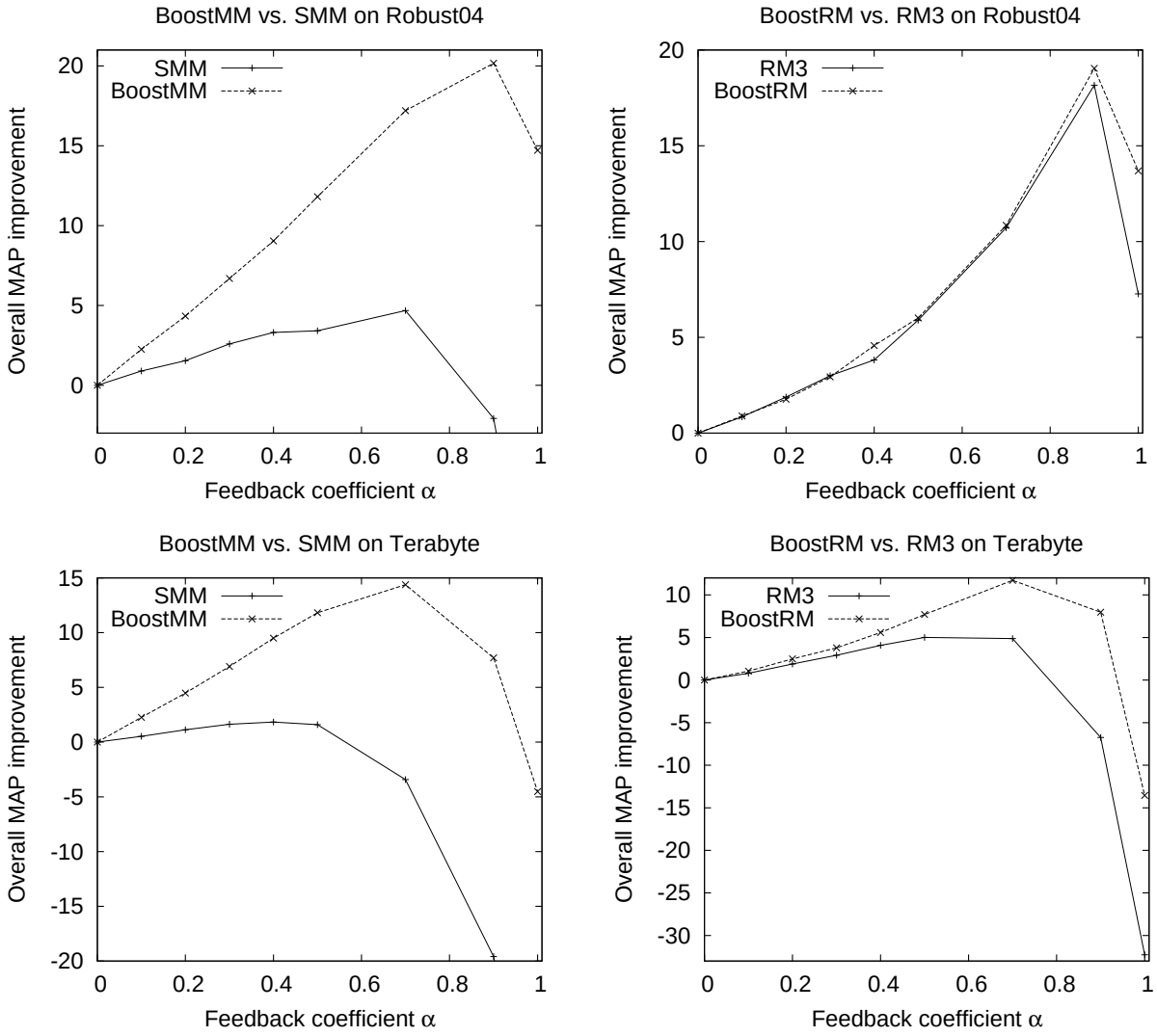

Figure 5.4: Sensitivity of the average improvement of MAP in all queries to the feedback coefficient α. SMM versus BoostMM and RM3 versus BoostRM are shown in the odd and even-number figures respectively.

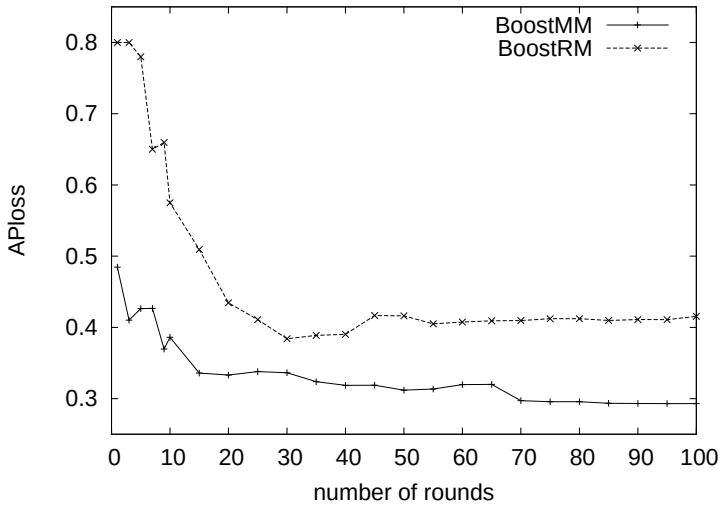

Figure 5.5: Sensitivity of BoostMM and BoostRM to the iteration number on Robust04 validate set.

methods to construct a larger set of basis feedback methods. Second, we are also interested in combining even more recently proposed pseudo feedback algorithms, e.g., [4, 103, 22], and other families of feedback methods, e.g., [88, 85], to diversify our basis feedback methods. Third, personalized search is another scenario which shares similar effectiveness-robustness tradeoff issues; thus it is also interesting to improve personalized search by exploring the FeedbackBoost framework.

Chapter 6

Lower-Bounding Term Frequency Normalization

6.1 Introduction

We have presented several more effective document and query language models in the previous Chapters. Besides document and query language models, the third key component in any language modeling approach is its scoring function for computing the relevance score of a document in response to a query. The language modeling approach scores a document mainly based on the query likelihood score [81], which has proven to be empirically effective for many retrieval tasks. However, after more than one decade of research, the basic query likelihood retrieval function still remains state-of-the-art [118], which shows that it is very difficult to improve the basic language modeling retrieval function. In fact, there are similar observations in the literature of other retrieval models. For example, Okapi BM25 [86, 87], the pivoted length normalization method [97], and the PL2 method [5], all have been proposed for more than ten years, but still remain strong baselines in the classic probabilistic retrieval model, vector space model, and the divergence from randomness model, respectively.

In order to further develop more effective functions, it is necessary to understand the deficiencies of the current retrieval functions [29]. For example, in [97], it was revealed that the traditional vector space model retrieves documents with probabilities different from their probabilities of relevance, and the analysis led to the pivoted normalization retrieval function which has been shown to be substantially more effective than the traditional vector space model. In this work, we reveal a previously unknown common deficiency of existing retrieval models, including the query likelihood retrieval function, in optimizing the TF normalization component and propose a general way to address this deficiency that can be applied to multiple state-of-the-art retrieval models to improve their retrieval accuracy.

Previous work [29] has shown that all the effective retrieval models tend to rely on a reasonable way to combine multiple retrieval signals, such as term frequency (TF), inverse document frequency (IDF), and document length. A major challenge in developing an effective retrieval model lies in the fact that multiple signals generally interact with each other in a complicated way. For example, document length normalization is to regularize the TF heuristic which, if applied alone, would have a tendency to overly reward long documents due to their high likelihood of matching a query term more times than a short document. On the other hand, document length normalization can also overly

penalize long documents [97, 29]. What is the best way of combining multiple signals has been a long-standing open challenge. In particular, a direct application of a sound theoretical framework such as the language modeling approach to retrieval does not automatically ensure that we achieve the optimal combination of necessary retrieval heuristics as shown in [29].

To tackle this challenge, formal constraint analysis was proposed in [29]. The idea is to define a set of formal constraints to capture the desirable properties of a retrieval function related to combining multiple retrieval signals. These constraints can then be used to diagnose the deficiency of an existing model, which in turn provides insight into how to improve an existing model. Such an axiomatic approach has been shown to be useful for motivating and developing more effective retrieval models [31, 32, 21].

In this section, we follow this axiomatic methodology and reveal a previously unknown common deficiency of the current retrieval models, including the query likelihood retrieval function, in their TF normalization component, and propose a general strategy to fix this deficiency in multiple state-of-the-art retrieval models. Specifically, we show that the normalized TF may approach zero when the document is very long, which often causes a very long document with a non-zero TF (i.e., matching a query term) to receive a score too close to or even lower than the score of a short document with a zero TF (i.e., not matching the corresponding query term). As a result, the occurrence of a query term in a very long document would not ensure that this document be ranked above other documents where the query term does not occur, leading to unfair over-penalization of very long documents. For example, for a query "computer virus", a long document matching both "computer" and "virus" can easily be ranked lower than a short document matching only "computer".

The root cause for this deficiency is that the component of TF normalization by document length is not lower-bounded properly, i.e., the score "gap" between the presence and absence of a query term could be infinitely close to zero or even negative. In order to diagnose this problem, we first propose two desirable constraints to capture the heuristic of lower-bounding TF in a formal way, so that it is possible to apply them to any retrieval function analytically. We then use constraint analysis to examine several representative retrieval functions and show that all these retrieval functions can only satisfy the constraints for a certain range of parameter values and/or for a particular set of query terms. Empirical results further show that the retrieval performance tends to be poor when the parameter is out of the range or the query term is not in the particular set.

Motivated by this understanding, we propose a general and efficient methodology for introducing a sufficiently large lower bound for TF normalization, which can be applied directly to current retrieval models. Constraint analysis shows analytically that the proposed methodology can successfully fix or alleviate the problem.

Our experimental results on multiple standard collections demonstrate that the proposed methodology, incurring almost no additional computational cost, can be applied to state-of-the-art retrieval functions, such as Okapi BM25 [86,

87], the query likelihood retrieval function [81, 121], and the PL2 method [5], to significantly improve their average precision, especially when queries are verbose. Due to its effectiveness, efficiency, and generality, the proposed methodology can work as a "patch" to fix or alleviate the problem in current retrieval models, in a plug-and-play way.

6.2 Related Work

Developing effective retrieval models is a long-standing central challenge in information retrieval. Many different retrieval models have been proposed and tested, such as vector space models [89, 97], classical probabilistic retrieval models [85, 36, 86, 87], language models [81, 121], and the divergence from randomness approach [5]; a few representative retrieval models will be discussed in detail in Section 6.3. In our work, we reveal and address a common "bug" of these retrieval models (i.e., TF normalization is not lower-bounded properly), and develop a general plug-and-play "patch" to fix or alleviate this bug.

Term frequency is the earliest and arguably the most important retrieval signal in retrieval models [87, 97, 81, 121, 96, 5, 29, 70]. The use of TF can be dated back to Luhn's pioneer work on automatic indexing [67]. It is widely recognized that linear scaling in term frequency puts too much weight on repeated occurrences of a term. Thus, TF is often upper-bounded through some sub-linear transformations [87, 97, 81, 121, 5, 21, 70] to prevent the contribution from repeated occurrences from growing too large. Particularly, in Okapi BM25 [86, 87], it is easy to show that there is a strict upper bound $(k_1 + 1)$ for TF normalization. However, the other interesting direction, *lower-bounding TF*, has not been well addressed before. Our recent work [71] appears to be the first study that notices the inappropriate lower-bound of TF in BM25 through empirical analysis, but there is no theoretic diagnosis of the problem. Besides, the approach proposed in [71] is not generalizable to lower-bound TF normalization in retrieval models other than BM25. In this section, we extend [71] to show analytically and empirically that lower-bounding TF is necessary for all representative retrieval models and develop a general approach to effectively lower-bound TF in these retrieval models.

Document length normalization also plays an important role in almost all existing retrieval models to fairly retrieve documents of all lengths [97, 29], since long documents tend to use the same terms repeatedly (higher TF). For example, both Okapi BM25 [86, 87] and the pivoted normalization retrieval model [97] use the pivoted length normalization schema [97], which uses the average document length as the pivoted length to coordinate the normalization effects for documents longer than this pivoted length and documents shorter than it. The PL2 model, a representative of the divergence from randomness models [5], also uses the average document length to control document length normalization. A common deficiency of all these existing length normalization methods is that they tend to force the normalized TF to approach zero when documents are very long. As a result, a very long document with a non-zero

Model	$G\left(c(t,Q)\right)$	$F\left(c(t,D),\|D\|,td(t)\right)$
BM25	$\frac{(k_3+1)\cdot c(t,Q)}{k_3+c(t,Q)}$	$\frac{(k_1+1)c(t,D)}{k_1(1-b+b\cdot\|D\|/avdl)+c(t,D)}\cdot\log\frac{N+1}{df(t)}$
PL2	$c(t,Q)$	$\begin{cases}\frac{tfn_t^D\cdot\log_2\left(tfn_t^D\cdot\lambda_t\right)+\log_2 e\cdot\left(1/\lambda_t-tfn_t^D\right)+0.5\log_2\left(2\pi\cdot tfn_t^D\right)}{tfn_t^D+1} & \text{if } tfn_t^D>0 \text{ and } \lambda_t>1\\ 0 & \text{otherwise}\end{cases}$
Dir	$c(t,Q)$	$\log\left(\frac{\mu}{\|D\|+\mu}+\frac{c(t,D)}{(\|D\|+\mu)p(t\|C)}\right)$
Piv	$c(t,Q)$	$\begin{cases}\frac{1+\log(1+\log(c(t,D)))}{1-s+s\cdot\|D\|/avdl}\cdot\log\frac{N+1}{df(t)} & \text{if } c(t,D)>0\\ 0 & \text{otherwise}\end{cases}$

Table 6.1: Document and query term weighting components of representative retrieval functions.

TF could receive a score too close to or even lower than the score of a short document with a zero TF, which is clearly unreasonable. Although some exiting studies have attempted to use a sub-linear transformation of document length (e.g., the squared root of document length [25]) to heuristically replace the original document length in length normalization, they are not guaranteed to solve the problem and often lose to standard document length normalization such as the pivoted length normalization in terms of retrieval accuracy. Our work aims at addressing this inherent weakness of traditional document length normalization in a more general and effective way.

Constraint analysis has been explored in information retrieval to diagnostically evaluate existing retrieval models [29, 30], introduce novel retrieval signals into existing retrieval models [104], and guide the development of new retrieval models [31, 21]. The constraints in these studies are basic and are designed mostly based on the analysis of some common characteristics of existing retrieval formulas. Although we also use constraint analysis, the proposed constraints are novel and are inspired by our empirical finding of a common deficiency of the existing retrieval models. Moreover, although some existing constraints (e.g., LNCs and TF-LNC in [29, 30]) are also meant to regularize the interactions between TF and document length, they tend to be loose and cannot capture the heuristic of lower-bounding TF normalization. For example, the modified Okapi BM25 satisfies all the constraints proposed in [29, 30], but it still fails to lower-bound TF normalization properly. In this sense, the proposed two new constraints are complimentary to existing constraints [29, 30].

6.3 Motivation of Lower-Bounding Term Frequency Normalization

In this section, we discuss and analyze a common deficiency (i.e., lack of appropriate lower bound for TF normalization) of four state-of-the-art retrieval functions, which respectively represent the classical probabilistic retrieval model (Okapi BM25 [86, 87]), the divergence from randomness approach (PL2 [5]), the language modeling approach (query likelihood with Dirichlet prior smoothing [121]), and the vector space model (pivoted normalization [97, 96]).

An effective retrieval function is generally comprised of two basic separable components: a within-query scoring formula for weighting the occurrences of a term in the query and a within-document scoring formula for weighting

Notation	Description
$c(t, D)$	Frequency of term t in document D
$c(t, Q)$	Frequency of term t in query Q
N	Total number of docs in the collection
$df(t)$	Number of documents containing term t
$td(t)$	Any measure of discrimination value of term t
$\|D\|$	Length of document D
$avdl$	Average document length
$c(t, C)$	Frequency of term t in collection C
$p(t\|C)$	Probability of a term t given by the collection language model [121]

Table 6.2: Notation

the occurrences of this term in a document. We will represent each retrieval function in terms of these two separable components to make it easier for us to focus on studying the document side weighting:

$$S(Q, D) = \sum_{t \in Q} G\left(c(t, Q)\right) \cdot F\left(c(t, D), |D|, td(t)\right) \tag{6.1}$$

where $S(Q, D)$ is the total relevance score assigned to document D with respect to query Q, and $G(\cdot)$ and $F(\cdot)$ are within-query scoring function and within-document scoring function respectively. In Table 6.1, we show how this general scheme can be used to represent all the four major retrieval models. Other related notations are listed in Table 6.2. Note that most of the notations were also used in some previous work, e.g., [29], and will be adopted throughout our work.

6.3.1 Deficiency of Existing Retrieval Functions

Okapi BM25 (BM25): The Okapi BM25 method [86, 87] is a representative retrieval function that represents the classical probabilistic retrieval model. The BM25 retrieval function is summarized in the second row of Table 6.1. Following work [29], we modify the original IDF formula of BM25 to avoid the problem of possibly negative IDF values. The within-document scoring function of BM25 can be re-written as follows:

$$F_{BM25}\left(c(t, D), |D|, td(t)\right) = \frac{(k_1 + 1) \cdot tfn_t^D}{k_1 + tfn_t^D} \cdot \log \frac{N + 1}{df(t)} \tag{6.2}$$

where k_1 is a parameter, and tfn_t^D is the normalized TF by document length using pivoted length normalization [97].

$$tfn_t^D = \frac{c(t, D)}{1 - b + b\frac{|D|}{avdl}} \tag{6.3}$$

where b is the slope parameter in pivoted normalization.

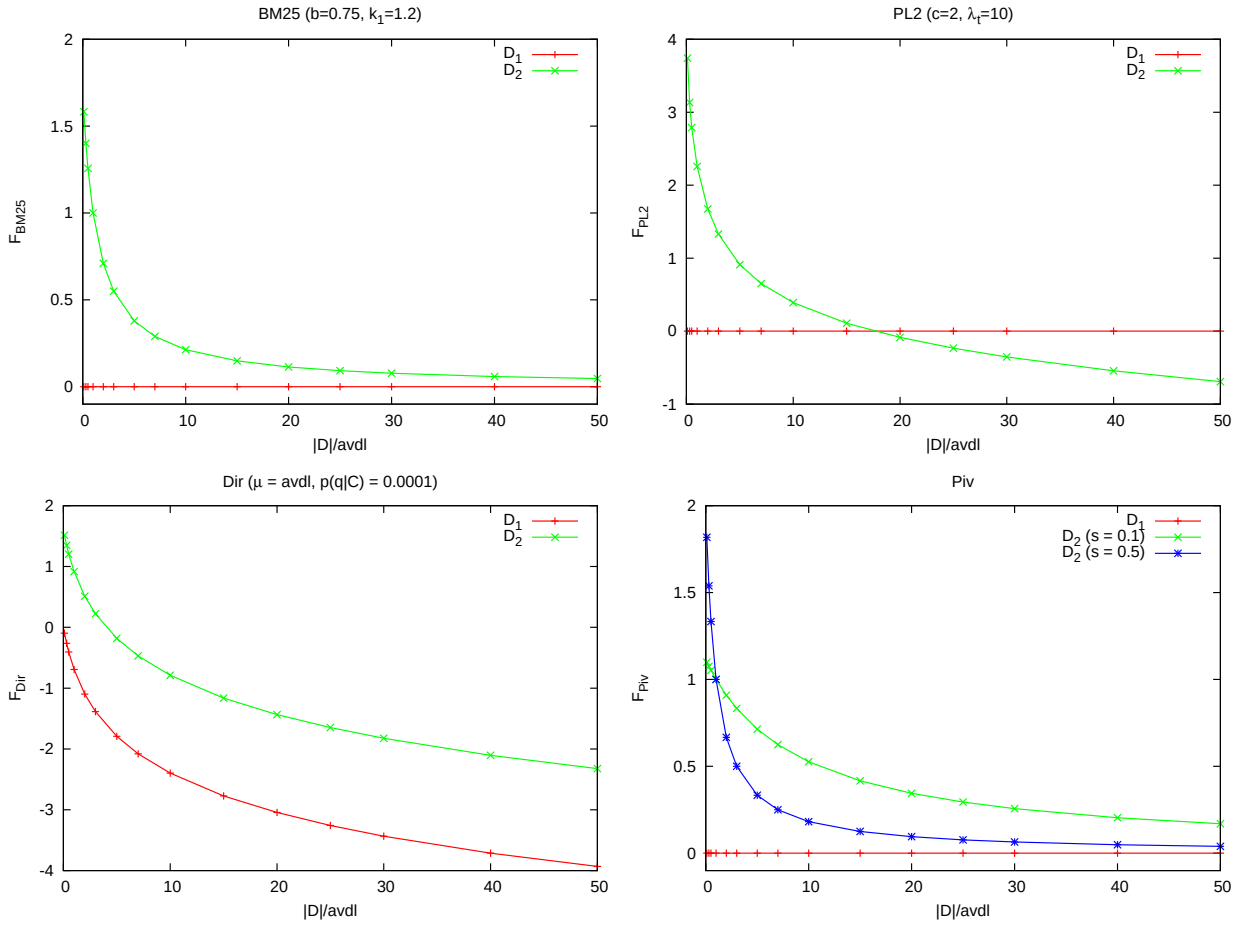

Figure 6.1: Comparison of the within-document term scores, i.e., $F(\cdot)$, of documents D_1 and D_2 w.r.t. query term t against different document lengths, where we assume $c(t, D_1) = 0$ and $c(t, D_2) = 1$. Here, x-axis and y-axis stand for the length of documents and the within-document term scores respectively.

When a document is very long (i.e., $|D|$ is much larger than $avdl$), we can see that tfn_t^D could be very small and approach 0. Consequently, F_{BM25} will also approach 0 as if t did not occur in D. It can be seen clearly in Figure 6.1 (1): when $|D_2|$ becomes very large, the score difference between D_2 and D_1 appears to be very small. This by itself, would not necessarily be a problem, but the problem is that, the occurrence of t in a very long document D fails to ensure D to be ranked above other documents where t does not occur. It suggests that the occurrences of a query term in very long documents may not be rewarded properly by BM25, and thus those very long documents could be overly penalized, which as we will show later, is indeed true.

PL2 Method (PL2): The PL2 method is a representative retrieval function of the divergence from randomness framework [5]. In this section, we use the modified PL2 formula derived by Fang et al. [30] instead of the original PL2 formula [5]. The only difference between this modified PL2 function and the original PL2 function is that the former essentially ignores non-discriminative query terms. It has been shown that the modified PL2 is more effective

and robust than the original PL2 [30]. The modified PL2 (still called PL2 for convenience in the following sections) is presented in the third row of Table 6.1, where $\lambda_t = \frac{N}{c(t,C)}$ is the term discrimination value, and tfn_t^D is the normalized TF by document length:

$$tfn_t^D = c(t, D) \cdot \log_2 \left(1 + c \cdot \frac{avdl}{|D|}\right) \tag{6.4}$$

where $c > 0$ is a retrieval parameter.

We can see that, when a document is very long, tfn_t^D could be very small and approach 0, which is very similar to the corresponding component in BM25. What is worse is that, when tfn_t^D is sufficiently small, the within-document score F_{PL2} will be a **negative** number surprisingly. However, as shown in Table 6.1, even if the term is missing, i.e., $c(t, D) = 0$, F_{PL2} can still receive a default *zero* score. This interesting observation is illustrated in Figure 6.1 (2). It suggests that a very long document that matches a query term may be penalized even more than another document (the length can be arbitrary) that does not match the term; consequently, those very long documents tend to be overly penalized by PL2.

Query Likelihood with Dirichlet Prior Method (Dir): The query likelihood with Dirichlet prior method is one of the best performing language modeling approaches [121]. It is presented in the fourth row of Table 6.1, where μ is the Dirichlet prior.

It is observed that, the within-document scoring function F_{Dir} is monotonically decreasing with the document length variable. And when a document D_2 is very long, say $50 * avdl$, even if it matches a query term, the within-document score of this term could still be arbitrarily small. And this score could be smaller than that of any average-length document D_1 which does not match the term. This is shown clearly in Figure 6.1 (3). Thus, the Dirichlet prior method can also overly penalize very long documents.

Pivoted Normalization Method (Piv): The pivoted normalization retrieval function [96, 29] represents one of the best performing vector space models. The detailed formula is shown in the last row of Table 6.1, where s is the slope parameter. Similarly, the analysis of the pivoted normalization method also shows that it tends to overly penalize very long documents, as shown in Figure 6.1 (4).

6.3.2 Empirical Evidence: Likelihood of Relevance/Retrieval

Our analysis above has shown that, *in principle*, all these retrieval functions tend to overly penalize very long documents. Now we turn to seeking empirical evidence to see if this common deficiency hurts document retrieval *in practice*.

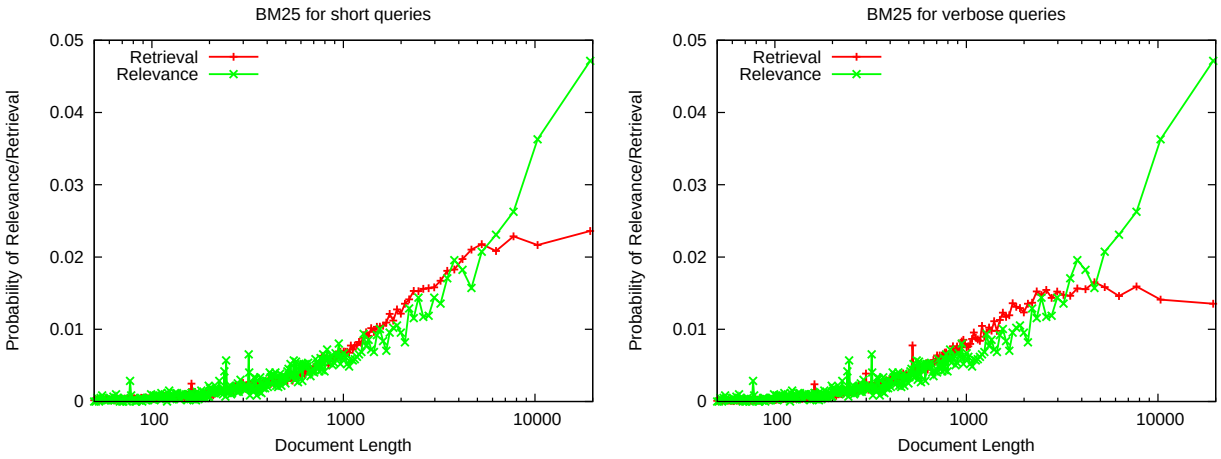

Figure 6.2: Comparison of retrieval and relevance probabilities against all document lengths when using BM25 on short (left) and verbose (right) queries.

Inspired by Singhal et al.'s finding that a good retrieval function should retrieve documents of all lengths with similar chances as their likelihood of relevance [97], we compare the retrieval pattern of different retrieval functions with the relevance pattern. We follow the binning analysis strategy proposed in [97] and plot the two patterns against all document lengths on WT10G in Figure 6.2, where the bin size is set to 5000. Due to the space reason, we only plot BM25 as an example. But it is observed that other retrieval functions have similar trends as BM25. The plot shows clearly that BM25 retrieves very long documents with chances much lower than their likelihood of relevance. This empirically confirms our previous analysis that very long documents tend to be overly penalized.

6.4 Formal Constraints

A critical question is thus how we can regulate the interactions between term frequency and document length when a document is very long so that we can fix this common deficiency of current retrieval models?

To answer this question, we first propose two desirable heuristics that any reasonable retrieval function should implement to properly lower bound TF normalization when documents are very long: (1) there should be a sufficiently large gap between the presence and absence of a query term, i.e., the effect of document length normalization should not cause a very long document with a non-zero TF to receive a score too close to or even lower than a short document with a zero TF; (2) a short document that only covers a very small subset of the query terms should not easily dominate over a very long document that contains many distinct query terms.

Next, in order to analytically diagnose the problem of over-penalizing very long documents, we propose two formal constraints to capture the above two heuristics of lower bounding TF normalization so that it is possible to apply them to any retrieval function analytically. The two constraints are defined as follows:

LB1: Let Q be a query. Assume D_1 and D_2 are two documents such that $S(Q, D_1) = S(Q, D_2)$. If we reformulate the query by adding another term $q \notin Q$ into Q, where $c(q, D_1) = 0$ and $c(q, D_2) > 0$, then $S(Q \cup \{q\}, D_1) < S(Q \cup \{q\}, D_2)$.

LB2: Let $Q = \{q_1, q_2\}$ be a query with two terms q_1 and q_2. Assume $td(q_1) = td(q_2)$, where $td(t)$ can be any reasonable measure of term discrimination value. If D_1 and D_2 are two documents such that $c(q_2, D_1) = c(q_2, D_2) = 0$, $c(q_1, D_1) > 0$, $c(q_1, D_2) > 0$, and $S(Q, D_1) = S(Q, D_2)$, then $S(Q, D_1 \cup \{q_1\} - \{t_1\}) < S(Q, D_2 \cup \{q_2\} - \{t_2\})$, for all t_1 and t_2 such that $t_1 \in D_1$, $t_2 \in D_2$, $t_1 \notin Q$ and $t_2 \notin Q$.

The first constraint LB1 captures the basic heuristic of 0-1 gap in TF normalization, i.e., the gap between presence and absence of a term should not be closed by document length normalization. Specifically, if a query term does not occur in document D_1 but occurs in document D_2, and both documents receive the same relevance score from matching other query terms, then D_1 should be scored lower than D_2, no matter what are the length values of D_1 and D_2. In other words, the occurrence of a query term in a very long document should still be able to differentiate this document from other documents where the query term does not occur.

In fact, when $F(0, |D|, td(t))$ is a document-independent constant, LB1 can be derived from a basic TF constraint, TFC1 [29]. Here, $F(0, |D|, td(t))$ is the document weight for a query term t not present in document D, i.e., $t \in Q$ but $t \notin D$. This property is presented below in Theorem 1.

Theorem 1 *LB1 is implied by the TFC1 constraint, if the within-document weight for any missing term is a document independent constant.*

Proof: Let Q be a query. Assume D_1 and D_2 are two documents such that $S(Q, D_1) = S(Q, D_2)$. We reformulate query Q by adding another term $q \notin Q$ into the query, where $c(q, D_1) = 0$ and $c(q, D_2) > 0$. If D_2' is another document, which is generated by replacing all the occurrences of q in D_2 with a non-query term $t \notin Q \cup \{q\}$, then $c(q, D_2') = 0$ and $S(Q, D_1) = S(Q, D_2) = S(Q, D_2')$. Due to the assumption that the document weight for the missing term q is a document independent constant, it follows that $S(Q \cup \{q\}, D_1) = S(Q \cup \{q\}, D_2')$. Finally, since $|D_2'| = |D_2|$ and $c(q, D_2') = 0 < c(q, D_2)$, according to TFC1, we get $S(Q \cup \{q\}, D_1) = S(Q \cup \{q\}, D_2') < S(Q \cup \{q\}, D_2)$. □

However, when the document weights for missing terms are document dependent, LB1 will not be redundant in the sense that it cannot be derived from other constraints such as the proposed LB2 and the seven constraints proposed in [29]. For example, the Dirichlet prior retrieval function, as shown in Table 6.1, has a document-dependent weighting function for a missing term, which is $\log \frac{\mu}{|D|+\mu}$. As will be shown later, the Dirichlet prior method violates LB1, although it satisfies LB2 and most of the constraints proposed in [29].

The second constraint LB2 states that if two terms have the same discrimination value, a repeated occurrence of one term is not as important as the first occurrence of the other. LB2 essentially captures the intuition that covering

more *distinct* query terms should be rewarded sufficiently, even if the document is very long. For example, given a query $Q = \{$"computer", "virus"$\}$, if two documents D_1 and D_2 with identical relevance scores with respect to Q both match "computer", but neither matches "virus", then if we add an occurrence of "virus" to D_1 to generate D_1' and add an occurrence of "computer" to D_2 to generate D_2', we should ensure that D_1' has a higher score than D_2'. This intuitively makes sense because D_1' is more likely to be related to computer virus, while D_2' may be just about other aspects of computer.

LB1 and LB2 are two necessary constraints to ensure that very long documents would not be overly penalized. When either is violated, the retrieval function would likely not perform well for very long documents and there should be room to improve the retrieval function through improving its ability of satisfying the corresponding constraint.

6.5 Constraint Analysis on Current Retrieval Models

Okapi BM25 (BM25): BM25 satisfies TFC1 [29], and the within-document weight for any missing term is always 0. Therefore, BM25 satisfies LB1 unconditionally according to Theorem 1.

We now examine LB2. Due to the sub-linear property of TF normalization, we only need to check LB2 in the case when $c(q_1, D_1) = 1$, since when $c(q_1, D_1) > 1$, it is even harder to violate the constraint. Consider a common case when $|D_1| = avdl$. It can be shown that the LB2 constraint is equivalent to the following constraint on $|D_2|$:

$$|D_2| < \left(\frac{2k_1 + 2}{(k_1)^2 \cdot b} + 1\right) \cdot avdl \tag{6.5}$$

This means that LB2 is satisfied only if $|D_2|$ is smaller than a certain upper bound. Thus, a sufficiently long document would violate LB2. Note that the upper bound of $|D_2|$ is a monotonically decreasing function with both b and k_1. This suggests that a larger b or k_1 would lead BM25 to violate LB2 more easily, which is confirmed by our experiments.

PL2 Method (PL2): In Fang et al.'s work [30], the TFC1 constraint is regarded equivalent to that "the first partial derivative of the formula w.r.t. the TF variable should be positive", which has been shown to be satisfied by the modified PL2 [30]. However, the PL2 function is not continuous when the TF variable is zero, and what is worse is that,

$$\lim_{c(t,D)\to 0} F_{PL2}\left(c(t,D), |D|, td(t)\right) < F_{PL2}\left(0, |D|, td(t)\right) = 0 \tag{6.6}$$

which shows that even the modified PL2 still fails to satisfy TFC1. So we cannot use Theorem 1 for PL2.

We thus check both LB1 and LB2 directly. Since the optimal setting of parameter c is usually larger than 1 [29], we consider a common case when $|D_1| = \frac{c}{3} \cdot avdl$. Similar to the analysis on BM25, we only need to examine LB2 for $c(q_1, D_1) = 1$. The LB1 constraint is approximately equivalent to

$$|D_2| < \frac{c}{2^{\exp\left(-\frac{2}{\lambda_t}-1.84\right)} - 1} \cdot avdl \tag{6.7}$$

and LB2 is approximately equivalent to

$$|D_2| < \frac{c}{2^{\exp\left(0.27\log(\lambda_t)-\frac{2.27}{\lambda_t}-2.26\right)} - 1} \cdot avdl \tag{6.8}$$

Due to space limit, we cannot show all the derivation details.

We can see that both LB1 and LB2 set an upper bound for document length, suggesting that a very long document would violate both LB1 and LB2. However the upper bound introduced by LB1 is always larger than that introduced by LB2. So we focus on LB2 in the following sections.

The upper bound of document length in LB2 is monotonically increasing with both c and λ_t. It suggests that, when c is very small, there is a serious concern that long documents would be overly penalized. On the other hand, a more discriminative term also violates the constraint more easily. These analyses are confirmed by our experiments.

Query Likelihood with Dirichlet Prior Method (Dir): With Dir, the within-document weight for a missing term is $\log \frac{\mu}{|D|+\mu}$, which is document dependent. So Theorem 1 is not applicable to the Dirichlet method. We thus need to examine LB1 and LB2 directly.

First, we only check LB1 at the point of $c(q, D_2) = 1$, which is the easiest case for LB1 to be violated. By considering the common case that $|D_1| = avdl$, the LB1 constraint is equivalent to the following constraint on $|D_2|$:

$$|D_2| < avdl + \frac{1}{p(q|C)}\left(1 + \frac{avdl}{\mu}\right) \tag{6.9}$$

It shows that the Dirichlet method can only satisfy LB1 if $|D_2|$ is smaller than a certain upper bound, suggesting again that a very long document would violate LB1. And this upper bound is monotonically decreasing with both $p(q|C)$ and μ. On the one hand, a non-discriminative (i.e., large $p(q|C)$) term q violates LB1 easily; for example, if $\mu \cdot p(q|C) = 1$, the upper bound appears to be as low as $(2 * avdl + \mu)$. Thus, the Dirichlet method would overly penalize very long documents more for verbose queries. On the other hand, a large μ would also worsen the situation according to Formula 6.9. These are all confirmed by our experimental results.

Next, we turn to check LB2, which is equivalent to

$$\frac{n+1+\mu \cdot p(q|C)}{n+\mu \cdot p(q|C)} < \frac{1+\mu \cdot p(q|C)}{\mu \cdot p(q|C)} \tag{6.10}$$

where $n \in \{1, 2, \cdots\}$. Interestingly, this inequality is always satisfied, suggesting that the Dirichlet method satisfies LB2 *unconditionally*. We thus expect that the Dirichlet method would have some advantages in the cases when other retrieval functions tend to violate LB2.

Pivoted Normalization Method (Piv) It is easy to show that the pivoted normalization method also satisfies LB1 unconditionally. We now examine LB2. Similar to the analysis on BM25, we only need to check LB2 in the case of $c(q_1, D_1) = 1$. By considering a common case when $|D_1| = avdl$, we see that LB2 is equivalent to the following constraint on $|D_2|$:

$$|D_2| < \left(\frac{0.899}{s} + 1\right) \cdot avdl \tag{6.11}$$

This means that LB2 is satisfied only if $|D_2|$ is smaller than a certain upper bound. And this upper bound is a monotonically decreasing function with s. So, in principle, a larger s would lead the pivoted normalization method to violate LB2 more easily, which can also explain why the optimal setting of s tends to be small [29]. Of course, if s is set to zero, LB2 would be satisfied, but that would be to turn off document length normalization completely, which would clearly lead to non-optimal retrieval performance.

6.6 A General Approach to Lower-Bounding TF Normalization

The analysis above shows analytically that all the state-of-the-art retrieval models would tend to overly penalize very long documents. In order to avoid overly penalizing very long documents, we need to lower-bound TF normalization to make sure that the "gap" of the within-document scores $F(c(t, D), |D|, td(t))$ between $c(t, D) = 0$ and $c(t, D) > 0$ is sufficiently large. However, we do not want that the addition of this new constraint changes the implementations of other retrieval heuristics in these state-of-the-art retrieval functions, because the implementations of existing retrieval heuristics in these retrieval functions have been shown to work quite well [29].

We propose a general heuristic approach to achieve this goal by defining an improved within-document scoring formula F' as shown in Equation 6.12, where δ is a pseudo TF value to control the scale of the TF lower bound, and l is a pseudo document length which is document-independent. In this new formula, a retrieval model-specific, but document-independent value $F(\delta, l, td(t)) - F(0, l, td(t))$ would serve as an ensured "gap" between matching and

missing a term: if $c(t,D) > 0$, the component of TF normalization by document length will be lower-bounded by such a document-independent value, no matter how large $|D|$ would be.

$$F'(c(t,D),|D|,td(t)) = \begin{cases} F(c(t,D),|D|,td(t)) + F(0,l,td(t)) & \text{if } c(t,D)=0 \\ F(c(t,D),|D|,td(t)) + F(\delta,l,td(t)) & \text{otherwise} \end{cases} \tag{6.12}$$

It is easy to verify that $F'(c(t,D),|D|,td(t))$ is able to satisfy all the basic retrieval heuristics [29] that are satisfied by $F(c(t,D),|D|,td(t))$: first, it is trivial to show that, if $F(c(t,D),|D|,td(t))$ satisfies TFCs, $F'(c(t,D),|D|,td(t))$ will also satisfy them; secondly, $F(\delta,l,td(t))$ and $F(0,l,td(t))$, as two special points of $F(c(t,D),|D|,td(t))$, satisfy the TDC constraint in exactly the same way as $F(c(t,D),|D|,td(t))$, so does $F'(c(t,D),|D|,td(t))$; finally, since the newly introduced components are document-independent, they raise no problem for LNCs and TF-LNC.

The proposed methodology is very efficient, as it only adds a retrieval model specific but document-independent value to those standard retrieval functions. For a query Q, we only need to calculate $|Q|$ such values, which can even be done offline. Therefore, our method incurs almost no additional computational cost.

Finally, we can obtain the corresponding lower-bounded retrieval function through substituting $F'(c(t,D),|D|,td(t))$ for $F(c(t,D),|D|,td(t))$ in each retrieval function,

Take BM25 as an example. Obviously $F'(0,|D|,td(t)) = 0$. In $F(\delta,l,td(t))$, since l is a constant document length variable used for document length normalization, its influence can be absorbed into the TF variable δ, we thus set $l = avdl$ simply. Then, we obtain $F(\delta,avdl,td(t)) = \frac{(k_1+1)\delta}{k_1+\delta}\log\frac{N+1}{df(t)}$. Clearly parameter k_1 can also be absorbed into δ, and the above formula is simplified again as $\delta\log\frac{N+1}{df(t)}$. Finally, we derive a lower-bounded BM25 function, namely **BM25+**, as shown in the following Formula 6.13.

$$\sum_{t\in Q\cap D}\frac{(k_3+1)c(t,Q)}{k_3+c(t,Q)}\times\left[\frac{(k_1+1)c(t,D)}{k_1\left(1-b+b\frac{|D|}{avdl}\right)+c(t,D)}+\delta\right]\times\log\frac{N+1}{df(t)} \tag{6.13}$$

$$\sum_{t\in Q\cap D}c(t,Q)\left[\log\left(1+\frac{c(t,D)}{\mu\cdot p(t|C)}\right)+\log\left(1+\frac{\delta}{\mu\cdot p(t|C)}\right)\right]+|Q|\cdot\log\frac{\mu}{|D|+\mu} \tag{6.14}$$

$$\sum_{t\in Q\cap D}c(t,Q)\left[\frac{1+\log\left(1+\log(c(t,D))\right)}{1-s+s\frac{|D|}{avdl}}+\delta\right]\log\frac{N+1}{df(t)} \tag{6.15}$$

$$\sum_{t \in Q \cap D, \lambda_t > 1} c(t, Q)$$
$$\times \left[\frac{tfn_t^D \log_2 \left(tfn_t^D \cdot \lambda_t \right) + \log_2 e \cdot \left(\frac{1}{\lambda_t} - tfn_t^D \right) + \frac{\log_2 \left(2\pi \cdot tfn_t^D \right)}{2}}{tfn_t^D + 1} + \frac{\delta \log_2 \left(\delta \cdot \lambda_t \right) + \log_2 e \cdot \left(\frac{1}{\lambda_t} - \delta \right) + \frac{\log_2 (2\pi\delta)}{2}}{\delta + 1} \right] \quad (6.16)$$

Similarly, we can derive a lower-bounded Dirichlet prior method (**Dir+**), a lower-bounded pivoted normalization method (**Piv+**), and a lower-bounded PL2 (**PL2+**), which are presented in Formulas 6.14, 6.15, and 6.16 respectively.

Next, we check LB1 and LB2 on these four improved retrieval functions.

Lower-Bounded BM25 (BM25+): It is trivial to verify that BM25+ still satisfies LB1 unconditionally. To examine LB2, we apply an analysis method that is consistent with our analysis for BM25 in Section 6.5. The LB2 constraint on BM25+ is equivalent to

$$\frac{k_1}{k_1 + 2} < \frac{(k_1 + 1) \cdot 1}{k_1 \left(1 - b + b \frac{|D_2|}{avdl} \right) + 1} + \delta \quad (6.17)$$

which can be shown to be satisfied unconditionally if

$$\delta \geq \frac{k_1}{k_1 + 2} \quad (6.18)$$

Clearly, if we set δ to a sufficiently large value, BM25+ is able to satisfy LB2 unconditionally, which is also confirmed in our experiments that BM25+ works very well when we set $\delta = 1$.

Lower-Bounded PL2 (PL2+): We only need to check LB2 on PL2+, since it is easier to violate than LB1. With a similar analysis strategy as used for analyzing PL2, the LB2 constraint on PL2+ is equivalent to

$$|D_2| < \frac{c \cdot avdl}{2^{\exp\left(\left(0.27 - \frac{2\delta}{\delta+1} \right) \log(\lambda_t) - \frac{2.27 - \frac{2}{\delta+1}}{\lambda_t} - 2.26 - g(\delta) \right)} - 1} \quad (6.19)$$

where $g(\delta) = \frac{(2\delta+1)\log\delta - 2\delta + \log(2\pi)}{\delta+1}$. Due to space limit, we cannot show all the derivation details in this section.

We can see that, given a δ, the right side of the Formula 6.19 (i.e., the upper bound of $|D_2|$) is minimized when $\lambda_t = \frac{2.27\delta + 0.27}{1.73\delta - 0.27}$. This suggests that, in contrast to PL2, the upper bound of $|D_2|$ is not monotonically decreasing with λ_t. This interesting difference is shown clearly in Figure 6.3. Thus, if we set δ to an appropriate value to make the minimum upper bound still large enough (e.g., larger than the length of the longest document), PL2+ would not violate LB2.

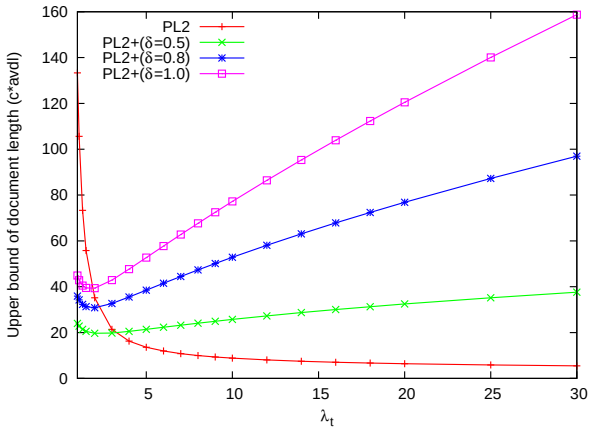

Figure 6.3: Comparison of upper bounds of document length in PL2 and PL2+ to satisfy LB2.

Lower-Bounded Query Likelihood with Dirichlet Method (Dir+): It is easy to show that Dir+ also satisfies LB2 unconditionally. We analyze Dir+ in the same way as analyzing Dir, and obtain the following equivalent constraint of LB1:

$$|D_2| < avdl + \frac{1+\delta}{p(t|C)}\left(1+\frac{avdl}{\mu}\right) + \frac{\delta}{\mu \cdot p^2(t|C)}\left(1+\frac{avdl}{\mu}\right) \tag{6.20}$$

We can see that, although Dir+ does not guarantee that LB1 is always satisfied, it indeed enlarges the upper bound of document length as compared to Dir in Section 6.5, and thus makes the constraint harder to violate. Generally, if we set δ to a sufficiently large value, the chance that very long documents are overly penalized would be reduced.

Lower-Bounded Pivoted Method (Piv+): It is easy to verify that Piv+ also satisfies LB1. Regarding LB2, similar to our analysis on Piv, the LB2 constraint on Piv+ is equivalent to

$$\log(1+\log 2) < \frac{1}{1-s+s\frac{|D_2|}{avdl}} + \delta \tag{6.21}$$

which is always satisfied if

$$\delta \geq \log(1+\log(2)) \approx 0.53 \tag{6.22}$$

This shows that Piv+ can be able to satisfy LB2 unconditionally with a sufficiently large δ.

	Terabyte	WT10G	Robust04	WT2G
queries	701-850	451-550	301-450 601-700	401-450
#qry(with qrel)	149	100	249	50
avg(ql_short)	3.13	4.24	2.74	2.46
avg(ql_verb)	11.55	11.61	15.47	13.86
#total_qrel	$28,640$	$5,981$	$17,412$	$2,279$
#documents	$25205k$	$1692k$	$528k$	$247k$
avdl	949	611	481	1056
std(dl)/avdl	2.63	2.31	1.19	2.14

Table 6.3: Document set characteristic

Query	Method	WT10G		WT2G		Terabyte		Robust04	
		MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
Short	BM25	0.1879	0.2898	0.3104	0.4840	0.2931	0.5785	0.2544	0.4353
	BM25+	**0.1962**^{4}	0.3040	0.3172^1	0.4820	**0.3004**^{1}	0.5685	**0.2553**	0.4357
	BM25+ ($\delta = 1.0$)	0.1927^3	0.3010	**0.3178**^{1}	0.4840	0.2997^4	0.5718	0.2548	0.4349
Verbose	BM25	0.1745	0.3250	0.2484	0.4380	0.2234	0.5221	0.2260	0.4036
	BM25+	**0.1850**^{1}	0.3360	**0.2624**^{3}	0.4400	0.2336^4	0.5309	0.2274	0.4056
	BM25+ ($\delta = 1.0$)	0.1841^1	0.3340	0.2565^1	0.4340	**0.2339**^{4}	0.5329	**0.2275**	0.4052

Table 6.4: Comparison of BM25 and BM25+ using cross validation. Superscripts 1/2/3/4 indicate that the corresponding MAP improvement is significant at the 0.05/0.02/0.01/0.001 level using the Wilcoxon test.

6.7 Experiments

6.7.1 Experimental Setup

We use four TREC collections: WT2G, WT10G, Terabyte, and Robust04, which represent different sizes and genre of text collections. WT2G, WT10G, and Terabyte are small, medium, and large Web collections respectively. Robust04 is a representative news dataset. We test two types of queries, short queries and verbose queries, which are taken from the title and the description fields of the TREC topics respectively. We use the Lemur toolkit and the Indri search engine (http://www.lemurproject.org/) to carry out our experiments. For all the datasets, the preprocessing of documents and queries is minimum, involving only Porter's stemming. An overview of the involved query topics, the average length of short/verbose queries, the total number of relevance judgments, the total number of documents, the average document length, and the standard deviation of document length in each collection are shown in Table 6.3.

We employ a 2-fold cross-validation for parameter tuning, where the query topics are split into even and odd number topics as the two folds. The top-ranked 1000 documents for each run are compared in terms of their mean average precisions (MAP), which also serves as the objective function for parameter training. In addition, the precision at top-10 documents (P@10) is also considered. Our goal is to see if the proposed general heuristic can work well for improving each of the four retrieval functions.

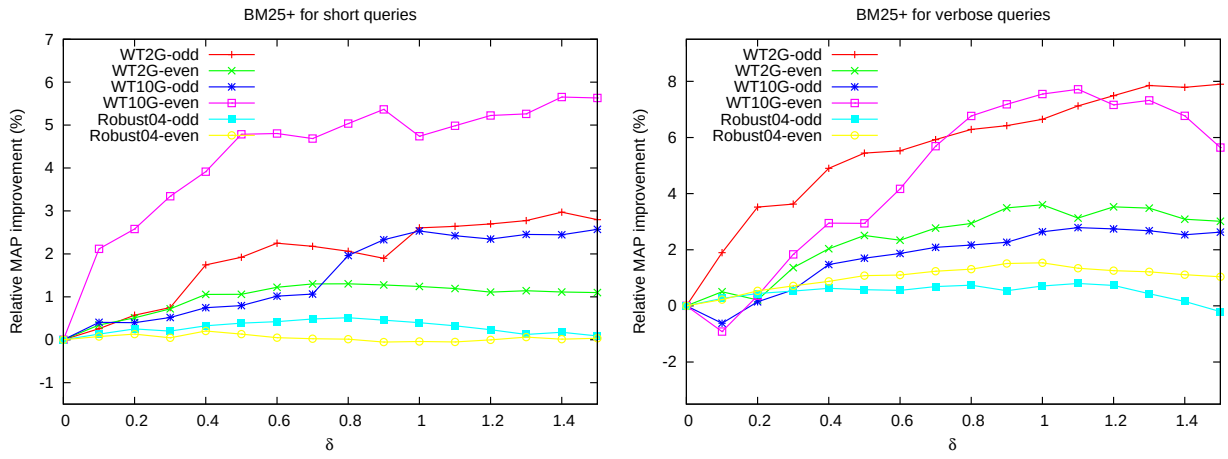

Figure 6.4: Performance Sensitivity to δ of BM25+, where y-axis shows the relative MAP improvements of BM25+ over BM25, and suffix '-even'/'-odd' indicates that only even/odd-number query topics are used.

6.7.2 Lower-Bounded BM25 (BM25+) VS. BM25

In both BM25+ and BM25, we train b and k_1 using cross validation, where b is tuned from 0.1 to 0.9 in increments of 0.1, and k_1 is tuned from 0.2 to 4.0 in increments of 0.2. Besides, in BM25+, parameter δ is trained using cross validation, where δ is tuned from 0.0 to 1.5 in increments of 0.1, but we also create a special run in which δ is fixed to 1.0 empirically (labeled as BM25+ ($\delta = 1.0$)). The comparison results of BM25+ and BM25 are presented in Table 6.4.

The results demonstrate that BM25+ outperforms BM25 consistently in terms of MAP and also achieves P@10 scores better than or comparable to BM25. The MAP improvements of BM25+ over BM25 are much larger on *Web* collections than on the *news* collection. In particular, the MAP improvements on all Web collections are statistically significant. This is likely because there are generally more very long documents in Web data, where the problem of BM25, i.e., overly-penalizing very long documents, would presumably be more severe. For example, Table 5.1 shows that the standard deviation of the document length is indeed larger on the three Web collections than on Robust04.

Another interesting observation is that, BM25+, even with a fixed $\delta = 1.0$, can still work effectively and stably across collections and outperform BM25 significantly. This empirically confirms the constraint analysis results in Section 6.6 that, when $\delta > \frac{k_1}{k_1+2}$, BM25+ can satisfy LB2 unconditionally. It thus suggests that the proposed constraints can even be used to guide parameter tuning.

We further plot the curves of MAP improvements of BM25+ over BM25 against different δ values in Figure 6.4, which demonstrates that, when δ is set to a value around 1.0, BM25+ works very well across all collections. Therefore, δ can be safely "eliminated" from BM25+ by setting it to a default value 1.0.

Regarding different query types, we observe that BM25+ improves more on verbose queries than on short queries.

Query	WT10G		WT2G		Terabyte		Robust04	
	b	k_1	b	k_1	b	k_1	b	k_1
Short	0.3	1.0	0.2	0.8	0.3	1.0	0.4	0.6
Verbose	0.6	2.0	0.6	1.6	0.4	1.8	0.7	1.2

Table 6.5: Optimal settings of b and k_1 in BM25.

For example, the MAP improvements on Web collections are often more than 5% for verbose queries and are around 2% for short queries. We hypothesize that BM25 may overly-penalize very long documents more seriously when queries are verbose, and thus there is more room for BM25+ to boost the performance. To verify our hypothesis, we collect the optimal settings of b and k_1 for BM25 in Table 6.5, which show that the optimal settings of b and k_1 are clearly *larger* for verbose queries than for short queries. Recall that our constraint analysis in Section 6.5 has shown that the likelihood of BM25 violating LB2 is monotonically increasing with parameters b and k_1. We can now conclude that BM25 indeed tends to overly penalize very long documents more when queries are more verbose.

So far we have shown that BM25+ is more effective than BM25, but if it is really because BM25+ has alleviated the problem of overly-penalizing very long documents? To answer this question, we plot the retrieval pattern of BM25+ as compared to the relevance pattern in a similar way as we have done in Section 6.3.2. The pattern comparison is presented in Figure 6.5. We can see that the retrieval pattern of BM25+ is more similar to the relevance pattern, especially for the retrieval of very long documents. This suggests that BM25+ indeed retrieves very long documents more fairly.

BM25+ and BM25 share two parameter b and k_1. We are also interested in understanding how these parameters affect the retrieval performance of BM25+ and BM25. So we first draw the sensitivity curves of BM25 and BM25+ to b in Figure 6.6. It shows that BM25+ is more robust than BM25; if we increase b, the performance of BM25 drops more quickly than BM25+. Next, we also plot the sensitivity curves of BM25 and BM25+ through varying k_1 in Figure 6.7. We can see that BM25 often does not work well when increasing k_1, while BM25+ appears more stable. These observations are not surprising, because increasing b or k_1 in BM25 could overly-penalize very long documents even more, as shown in our constraint analysis in Section 6.5.

6.7.3 Lower-Bounded PL2 (PL2+) VS. PL2

In both PL2+ and PL2, we train parameter c using cross validation, where c is tuned from 0.5 to 25 (27 values). Besides, in PL2+, parameter δ is also trained using cross validation, where δ is tuned from 0.0 to 1.5 in increments of 0.1. Also we create a special run of PL2+ in which δ is fixed to 0.8 empirically without training. The comparison results of PL2+ and PL2 are presented in Table 6.6.

The results show that PL2+ outperforms PL2 consistently, and even if we fix $\delta = 0.8$, PL2+ can still achieve stable

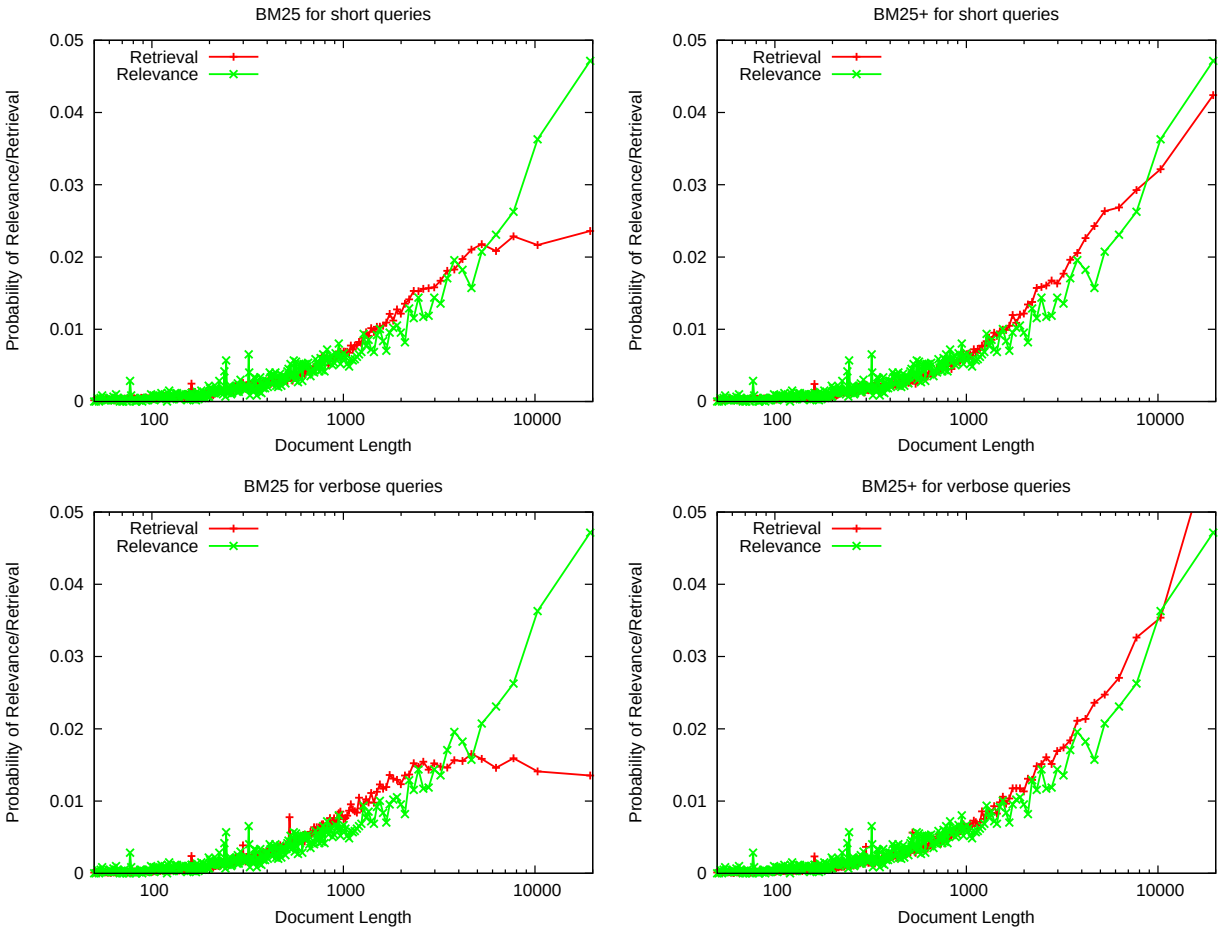

Figure 6.5: Comparison of retrieval and relevance probabilities against all document lengths when using BM25 (left) and BM25+ (right) for retrieval. It shows that BM25+ alleviates the problem of BM25 that overly penalizes very long documents.

improvements over PL2. Specifically, PL2+ improves significantly over PL2 for about **10**% on verbose queries, yet it only improves slightly on short queries; PL2+ appears to be less sensitive to the genre of collections, since it also improves significantly over PL2 on *news* data (verbose queries). We hypothesize that, PL2 may overly-penalize very long documents seriously on verbose queries but works well on short queries, and thus there is more room for PL2+ to improve the performance on verbose queries than on short queries. To verify it, we collect the optimal settings of c in PL2 and show them in Table 6.7. We can see that the optimal settings of c are "huge" for short queries as compared to that for verbose queries, presenting an obvious contrast. As a result, recalling the upper bound of document length in Formula 6.8, verbose queries would be more likely to violate LB2 even if a document is not very long (e.g., a news article), while short queries would only have a very small chance to violate LB2 even if a document is very long. Again, we can see that our constraint analysis is consistent with empirical results.

We further do a sensitivity test for both PL2 and PL2+ to parameter c. It shows that PL2+ works more robustly,

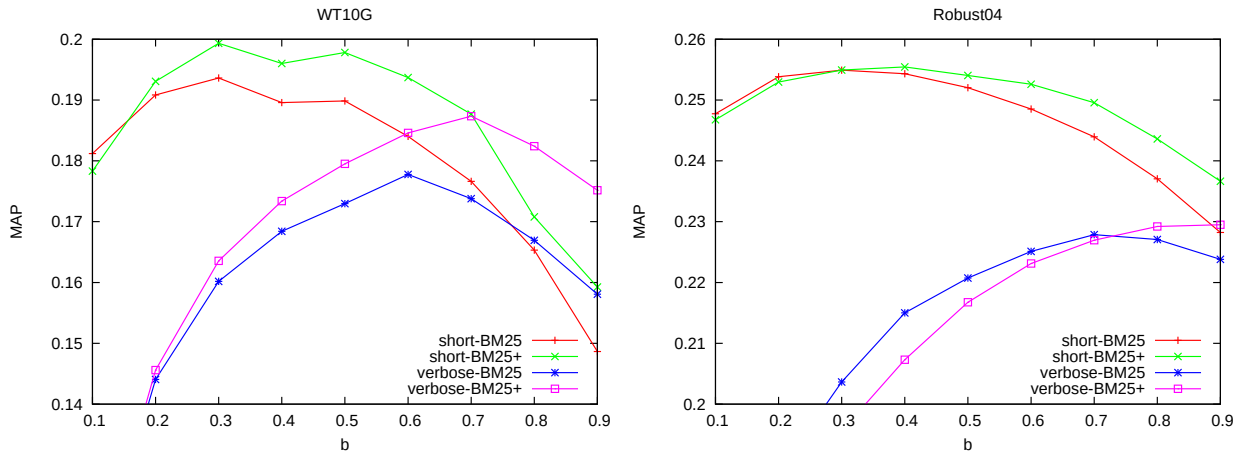

Figure 6.6: Performance Sensitivity to b of BM25 and BM25+ on WT10G (left) and Robust04 (right), where δ is fixed to 1.0 in BM25+ and k_1 is well tuned in both methods for different b values.

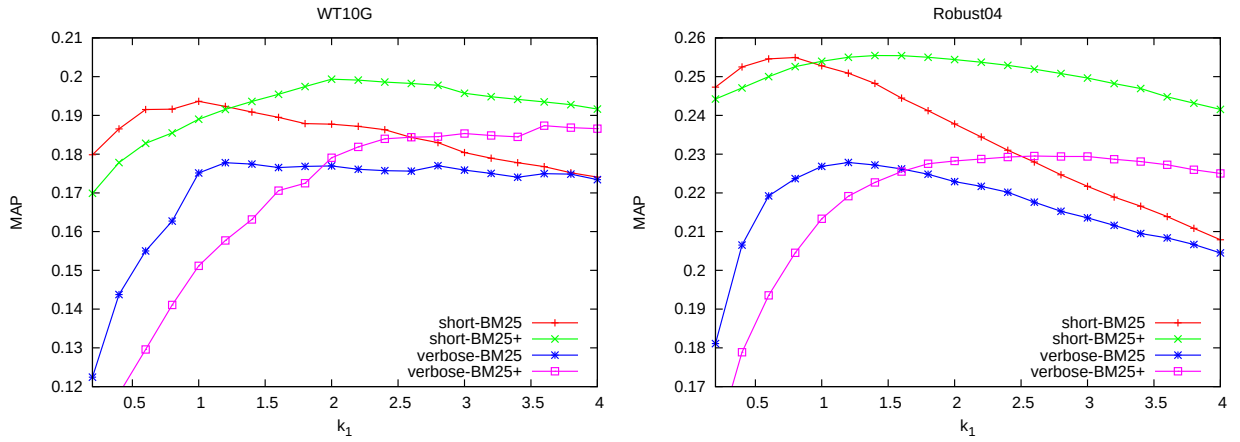

Figure 6.7: Performance Sensitivity to k_1 of BM25 and BM25+ on WT10G (left) and Robust04 (right), where δ is fixed to 1.0 in BM25+ and b is well tuned in both methods for different k_1 values.

especially when c is small, and that the performance of PL2 drops quickly when c becomes small, due to the violation of LB2.

6.7.4 Lower-Bounded Query Likelihood (Dir+) VS. Query Likelihood (Dir)

In both Dir+ and Dir, we train parameter μ using cross validation, where μ is tuned in a parameter space of 12 values from 500 to 10000. Besides, in Dir+, parameter δ is also trained, the candidate values of which are from 0.0 to 0.15 in increments of 0.01. Similarly, we also create a special run in which δ is fixed to 0.05 empirically without training. The comparison of Dir+ and Dir is presented in Table 6.8.

Overall, we observe that Dir+ improves over Dir consistently and significantly across different collections, and

Query	Method	WT10G	WT2G	Robust04
Short	PL2	0.1883	0.3231	0.2531
	PL2+	**0.1920**	0.3227	**0.2549**
	PL2+ ($\delta = 0.8$)	0.1912	**0.3255**	0.2540
Verbose	PL2	0.1695	0.2473	0.2185
	PL2+	**0.1886**^{4}	0.2595	**0.2348**^{4}
	PL2+ ($\delta = 0.8$)	**0.1886**^{4}	**0.2639**^{2}	0.2347^{4}

Table 6.6: Comparison of PL2 and PL2+ using cross validation. Superscripts 1/2/3/4 indicate that the corresponding MAP improvement is significant at the 0.05/0.02/0.01/0.001 level using the Wilcoxon test.

Query	WT10G	WT2G	Robust04
Short	9	23	9
Verbose	2	3	2

Table 6.7: Optimal settings of c in PL2.

Query	Method	WT10G	WT2G	Robust04
Short	Dir	0.1930	0.3088	0.2521
	Dir+	0.1961	0.3112^{2}	**0.2530**^{1}
	Dir+ ($\delta = 0.05$)	**0.1967**^{1}	**0.3123**^{3}	0.2525
Verbose	Dir	0.1790	0.2742	0.2329
	Dir+	**0.1874**^{3}	0.2867^{1}	0.2440^{4}
	Dir+ ($\delta = 0.05$)	0.1871^{3}	**0.2871**^{2}	**0.2440**^{4}

Table 6.8: Comparison of Dir and Dir+ using cross validation. Superscripts 1/2/3/4 indicate that the corresponding MAP improvement is significant at the 0.05/0.02/0.01/0.001 level using the Wilcoxon test.

Query	WT10G	WT2G	Robust04
Short	4000	2500	1000
Verbose	7000	8000	3000

Table 6.9: Optimal settings of μ in Dir.

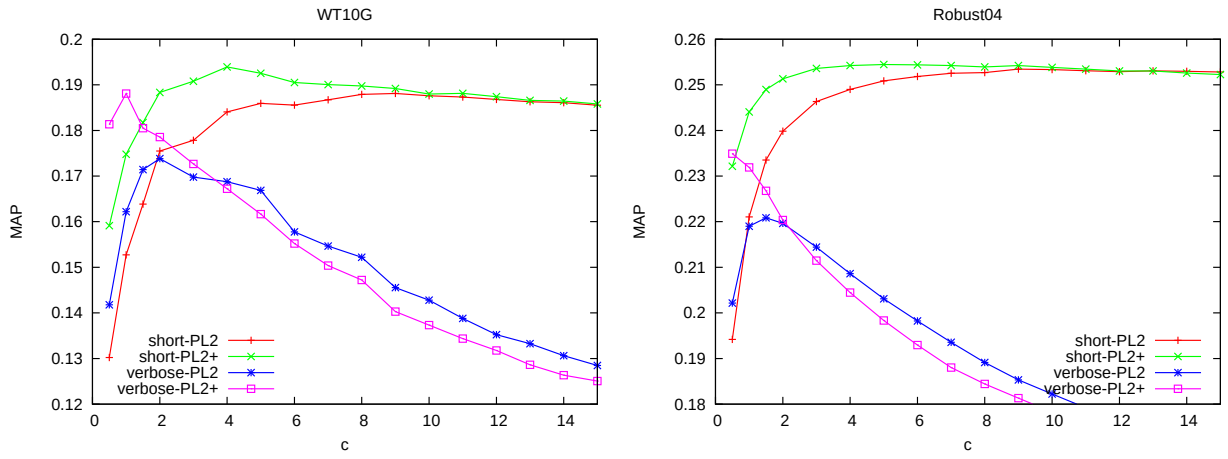

Figure 6.8: Performance Sensitivity to parameter c of PL2 and PL2+ on WT10G and Robust04, where δ is fixed to 0.8 in PL2+.

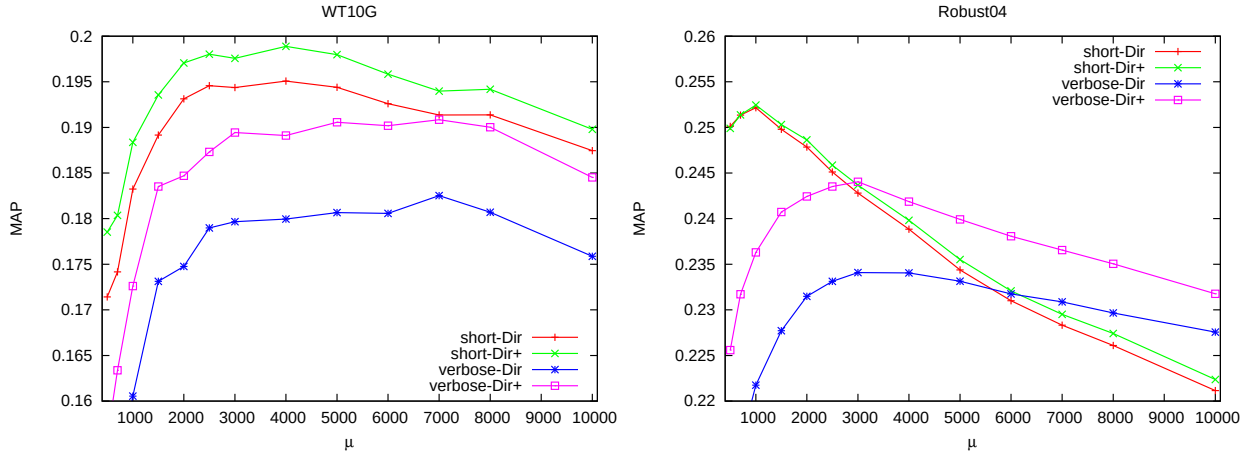

Figure 6.9: Performance Sensitivity to parameter μ of Dir and Dir+ on WT10G and Robust04, where δ is fixed to 0.05 in Dir+.

even if we fix $\delta = 0.05$ without training, Dir+ can still outperform Dir significantly in most cases. Note that, similar to BM25+ and PL2+, Dir+ works more effectively on verbose queries, which is consistent with our constraint analysis that Dir is more likely to overly penalize very long documents when a query contains more non-discriminative terms. Besides, we also collect the optimal settings of μ for Dir in Table 6.9, which show that the optimal settings of μ are also clearly *larger* for verbose queries than for short queries. Recall that our constraint analysis in Section 6.5 has shown that the likelihood of Dir violating LB1 is monotonically increasing with parameter μ. We can now conclude that Dir indeed tends to overly penalize very long documents more when queries are more verbose.

In addition, we further compare Dir+ and Dir thoroughly by varying μ from 500 to 10000, as shown in Figure 6.9. It shows that Dir+ is consistently better than Dir no matter how we change the μ value.

Moreover, comparing Table 6.8 with Table 6.4 and 6.6, we can see that Dir works clearly better on verbose queries

Query	Method	WT10G	WT2G	Robust04
Short	Piv	0.1870	0.2915	0.2410
	Piv+	0.1869	0.2945[1]	0.2455[1]
Verbose	Piv	0.1493	0.2148	0.2144
	Piv+	0.1493	0.2154	0.2150

Table 6.10: Comparison of Piv and Piv+ using cross validation. Superscripts 1 indicates that the corresponding MAP improvement is significant at the 0.05 level using the Wilcoxon test.

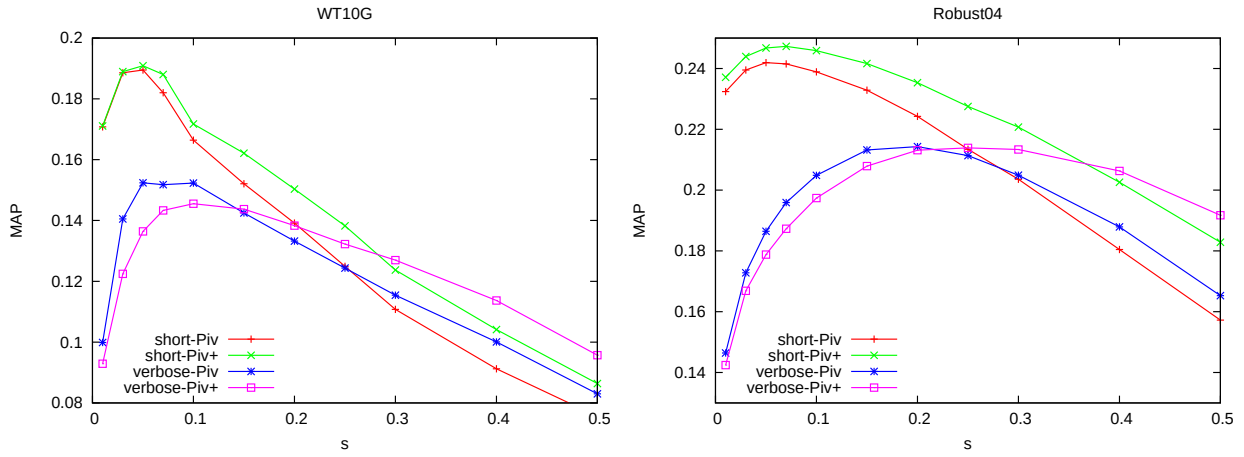

Figure 6.10: Performance Sensitivity to parameter s of Piv and Piv+ on WT10G and Robust04, where δ is fixed to 0.6 in Piv+.

than BM25 and PL2. One possible explanation is that Dir satisfies LB2 unconditionally, but BM25 and PL2 do not.

6.7.5 Lower-Bounded Piv (Piv+) VS. Piv

In both Piv+ and Piv, we train s using cross-validation, where s is tuned from 0.01 to 0.25 in increments of 0.02. Besides, in Piv+, parameter δ is also trained, the candidate values are from 0.0 to 1.5 in increments of 0.1. The comparison results of Piv+ and Piv are presented in Table 6.10.

Unfortunately, Piv+ does not improve over Piv significantly in most of the cases, which, however, is also as we expected: although there is an upper bound of document length for Piv to satisfy LB2 (as shown in Formula 6.11), this upper bound is often very large because the optimal setting of parameter s is often very small as presented in Table 6.11. Nevertheless, Piv+ would work much better than Piv when s is large, as observed in Figure 6.10.

Our experiments demonstrate empirically that, the proposed general methodology can be applied to state-of-the-art

Query	WT10G	WT2G	Robust04
Short	0.05	0.01	0.05
Verbose	0.05	0.11	0.19

Table 6.11: Optimal settings of s in Piv.

retrieval functions to successfully fix or alleviate their problem of overly-penalizing very long documents.

We have derived three effective retrieval functions, BM25+ (Formula 6.13), PL2+ (Formula 6.16), and Dir+ (Formula 6.14). All of them are as efficient as but more effective than their corresponding standard retrieval functions, i.e., BM25, PL2, and Dir, respectively. There is an extra parameter δ in the derived formulas, but we can set it to some default values (i.e., $\delta = 1.0$ for BM25+, $\delta = 0.8$ for PL2+, and $\delta = 0.05$ for Dir+), which perform quite well. The proposed retrieval functions can potentially replace its corresponding standard retrieval functions in all retrieval applications.

6.8 Summary

In this paper, we reveal a common deficiency of the current retrieval models: the component of term frequency (TF) normalization by document length is not lower-bounded properly; as a result, very long documents tend to be overly-penalized. In order to analytically diagnose this problem, we propose two desirable formal constraints to capture the heuristic of lower-bounding TF, and use constraint analysis to examine several representative retrieval functions. We find that all these retrieval functions can only satisfy the constraints for a certain range of parameter values and/or for a particular set of query terms. Empirical results further show that the retrieval performance tends to be poor when the parameter is out of the range or the query term is not in the particular set. To solve this common problem, we propose a general and efficient method to introduce a sufficiently large lower bound for TF normalization which can be shown analytically to fix or alleviate the problem.

Our experimental results on standard collections demonstrate that the proposed methodology, incurring almost no additional computational cost, can be applied to state-of-the-art retrieval functions, such as Okapi BM25 [86, 87], language models [121], and the divergence from randomness approach [5], to significantly improve the average precision, especially for verbose queries. Our work has also helped reveal interesting differences in the behavior of these state-of-the-art retrieval models. Due to its effectiveness, efficiency, and generality, the proposed methodology can work as a "patch" to fix or alleviate the problem in current retrieval models, in a plug-and-play way.

Chapter 7

Query Likelihood with Negative Query Generation

7.1 Introduction

In the previous chapter, we have identified and bridged an empirical theory-effectiveness gap in the query likelihood retrieval function using a *heuristic* approach, i.e., lower-bounding TF normalization. Although working effectively, such a heuristic approach tends to be inconsistent with the language modeling framework, the essence of which is statistical model estimation. In this chapter, we reveal and address a theoretical gap in the query likelihood method using a *probabilistic* approach; as a by-product, interestingly, the proposed approach also provides a probabilistic interpretation for the heuristic lower-bounding approach.

Although query likelihood has sound statistical foundation and performs well empirically, there was criticism about its theoretical foundation as a retrieval function [83, 48, 49]. In particular, Sparck Jones questioned "where is relevance?" [48]. Responding to this criticism, Lafferty and Zhai [59] showed that under some assumptions the query likelihood retrieval method can be justified based on probability ranking principle [84] which is regarded as the foundation of probabilistic retrieval models.

However, from theoretical perspective, the justification of using query likelihood as a retrieval function based on the probability ranking principle [59] requires an unrealistic assumption about the generation of a "negative query" from a document, which states that the probability that a user who dislikes a document would use a query does not depend on the particular document. This assumption enables ignoring the negative query generation in justifying using the standard query likelihood method as a retrieval function. In reality, however, this assumption does not hold because a user who dislikes a document would more likely avoid using words in the document when posing a query. This suggests that the standard query likelihood function is a potentially non-optimal retrieval function.

In order to address this theoretical gap between the standard query likelihood and the probability ranking principle, in this chapter, we attempt to bring back the component of negative query generation.

A main challenge in estimating the negative query generation component is to develop a *general* method for *any* document with respect to *any* query. Our solution to this problem is to estimate the probability of negative query generation purely based on document D so as to make it possible to incorporate the negative query generation

component when retrieving any document. Specifically, we exploit document D to infer the queries that a user would use to avoid retrieving D based on the intuition that such queries would not likely have any information overlap with D. We then propose an effective approach to estimate probabilities of negative query generation based on the principle of maximum entropy [45], which leads to a document-dependent negative query generation component that can be computed efficiently. Finally, we derive a more complete query likelihood retrieval function that also contains the negative query generation component, which essentially scores a document with respect to a query according to the ratio of the probability that a user who likes the document would pose the query to the probability that a user who dislikes the document would pose the query.

Similar to the standard query likelihood, a major deficiency of the proposed query likelihood with negative query generation is that it cannot easily incorporate query language models. To solve this problem, we further develop a more general probabilistic distance retrieval method, inspired by the development of the KL-divergence retrieval method [58]. With this method, we first estimate a regular document language model, a regular query language model, and a "negative document language model" based on the probabilities of negative query generation, and we then score a document with respect to a query based on the relative KL-divergence between the query language model and the corresponding document language model and between the query language model and the corresponding negative document language model. With this probabilistic distance retrieval method, Feedback can also be naturally cast as to improve the estimate of the query language model based on the feedback information. Interestingly, this probabilistic distance retrieval method covers the proposed query likelihood model with negative query generation as its special case.

Moreover, the developed query likelihood with negative query generation leads to the same ranking formula as derived by lower-bounding the query likelihood scoring function in the previous chapter, thus essentially providing a probabilistic interpretation for the heuristic method of lower-bounding term frequency normalization in the basic query likelihood method. Meanwhile, this connection also shows that the proposed query likelihood retrieval function with negative query generation improves over the basic query likelihood method empirically. Overall, the proposed approaches not only bridge the theoretical gap between the standard query likelihood and the probability ranking principle, but also improve retrieval effectiveness significantly.

7.2 Negative Query Generation

As discussed in Section 2.3, with query generation, the odds ratio of the probability ranking principle ends up with the following ranking formula:

$$O(R = \ell|Q, D) \propto \frac{p(Q|D, R = \ell)}{p(Q|D, R = \bar{\ell})} \tag{7.1}$$

There are two components in this model. $p(Q|D, R = \ell)$ can be interpreted as a positive query generation model. It is essentially the basic query likelihood, which suggests that the query generation probability used in all the query likelihood scoring methods intuitively means the probability that a user who likes document D would pose query Q. Another component $p(Q|D, R = \bar{\ell})$ can be interpreted as the generation probability of a "negative query" from a document, i.e., the probability that a user who dislikes a document D would use a query Q.

However, in order to justify using the basic query likelihood alone as the ranking formula, an unrealistic assumption has to be made about this negative query generation component, which states that the probability that a user who dislikes a document would use a query does not depend on the particular document [59], formally

$$p(Q|D, R = \bar{\ell}) = p(Q|R = \bar{\ell}) \tag{7.2}$$

This assumption enables ignoring the negative query generation in the derivation of the basic query likelihood retrieval function, leading to the following basic query likelihood scoring method: $O(R = \ell|Q, D) \propto p(Q|D, R = \ell) = P(Q|\theta_D)$.

In reality, however, this assumption does *not* hold because a user who dislikes a document would more likely avoid using words in the document when posing a query, suggesting a theoretical gap between the standard query likelihood and the probability ranking principle. These observations and analysis show that, although a standard statistical approach, the basic query likelihood function is a potentially non-optimal retrieval function.

In the following section, we attempt to improve the basic query likelihood function by estimating, rather than ignoring the component of negative query generation $p(Q|D, R = \bar{\ell})$.

7.3 Query Likelihood with Negative Query Generation

7.3.1 Negative Document Language Models

What would a user like if he/she does not like D? We assume that there exists a "complement document" $\bar{D}$, and that if a user does not like D, the user would like $\bar{D}$. That is, when generating query Q, if a user does not like D, the user

would randomly pick a word from $\bar{D}$. Formally,

$$p(w|D, R = \bar{\ell}) = p(w|\theta_{\bar{D}}) \tag{7.3}$$

The challenge now lies in how to estimate a language model $\theta_{\bar{D}}$, which we refer to as the "negative document language model" of D. Note that the negative document language model in our paper is still a "document" model, which is completely different from the relevance model $p(w|R = \ell)$ [61] and the irrelevance model $p(w|R = \bar{\ell})$ [109] that capture the probability of observing a word w relevant and non-relevant to a particular information need respectively.

Ideally we should use many actual queries by users who do not want to retrieve document D to estimate the probability $p(w|\theta_{\bar{D}})$. For example, we may assume that if a user sees a document in search results but does not click on it, he/she dislikes the document. Under this assumption, we can use all the queries from the users who "dislike" the document to approximate $\bar{D}$. However, in practice, only very few search results will be shown to users and certainly there are always queries that we would not even have seen. Yet, as a general retrieval model, the proposed method must have some way to estimate $\theta_{\bar{D}}$ for any document with respect to any query.

One straightforward way is using the background language model $p(w|C)$ to approximate $p(w|\theta_{\bar{D}})$, based on the intuition that almost all other documents in the collection are complementary to D:

$$p(w|\theta_{\bar{D}}) = p(w|C) \tag{7.4}$$

With this estimate of $p(w|\theta_{\bar{D}})$, the negative query generation component does not affect the ranking of documents, because the probability of negative query generation will be constant for all documents: it justifies the document independent negative query generation component in the standard query likelihood method. However, the content of document D is ignored in this estimate.

We are interested in estimating $p(w|\theta_{\bar{D}})$ in a general way based on the content of document D so as to make it possible to incorporate a document dependent negative query generation component when retrieving any document. Our idea is based on the intuition that if a user wants to avoid retrieving document D, he/she would more likely avoid using words in the document when posing a query. That is, the user would like a document $\bar{D}$ with little information overlap with D. Therefore, $\bar{D}$ should contain a set of words that do not exist in D, because given only document D available, the sole constraint of $\bar{D}$ is that, if a word w occurs in D, i.e., $c(w, D) > 0$, this word should not occur in $\bar{D}$.

$$c(w, \bar{D}) = \begin{cases} 0 & \text{if } c(w, D) > 0 \\ ? & \text{otherwise} \end{cases} \tag{7.5}$$

where "?" means unknown.

How to determine the count of a word in $\bar{D}$ if it does not occur in D? As the probability distribution of such a word is unknown, according to the *principle of maximum entropy* [45], each such a word occurring in $\bar{D}$ should have the same frequency $\delta > 0$, which maximizes the information entropy under the only prior data D. That is, $\bar{D}$ contains a set of words that are complementary to D in the universe word space (i.e., the whole word vocabulary V). Formally,

$$c(w, \bar{D}) = \begin{cases} 0 & \text{if } c(w, D) > 0 \\ \delta & \text{otherwise} \end{cases} \tag{7.6}$$

According to the maximum likelihood estimator, we have the following estimation of the document language model $\theta_{\bar{D}}$ for the multinomial model:

$$p_{ml}(w|\theta_{\bar{D}}) = \frac{c(w, \bar{D})}{|\bar{D}|} \tag{7.7}$$

where $|\bar{D}|$ is the "document" length of $\bar{D}$, which can be computed by aggregating frequencies of all words occurring in $\bar{D}$. Because the number of unique words in D is usually much smaller than the number of unique words in the whole document collection C (i.e., $|V|$), the number of unique words in $\bar{D}$ is approximately the same as $|V|$. Thus

$$|\bar{D}| = \sum_{w \in V} c(w, \bar{D}) \approx \delta |V| \tag{7.8}$$

Due to the existence of zero probabilities, $p_{ml}(w|\theta_{\bar{D}})$ needs smoothing. Following the estimation of regular document language models, we also choose the Dirichlet prior smoothing method due to its effectiveness in information retrieval [121]. Formally,

$$p(w|\theta_{\bar{D}}) = \frac{\delta |V|}{\delta |V| + \mu} p_{ml}(w|\theta_{\bar{D}}) + \frac{\mu}{\delta |V| + \mu} p(w|C) \tag{7.9}$$

where μ is the Dirichlet prior. Since the influence of μ can be absorbed into variable δ clearly, we thus set it simply to the same Dirichlet prior value as used for smoothing the regular document language model (see Equation 2.7 in Chapter 2).

7.3.2 Bringing Back the Negative Query Generation Component

Now we can bring back the negative query generation component to the query generation process based on the probability ranking principle:

$$\begin{aligned} O(R=\ell|Q,D) &\stackrel{rank}{=} \log \frac{p(Q|D,R=\ell)}{p(Q|D,R=\bar{\ell})} \\ &= \log p(Q|D,R=\ell) - \log p(Q|D,R=\bar{\ell}) \\ &= \log p(Q|\theta_D) - \log p(Q|\theta_{\bar{D}}) \end{aligned} \tag{7.10}$$

The negative query loglikelihood $\log p(Q|\theta_{\bar{D}})$ can be further written as

$$\log p(Q|\theta_{\bar{D}}) \stackrel{rank}{=} -\sum_{w \in Q \cap D} c(w,Q) \log \left(1 + \frac{\delta}{\mu p(w|C)}\right) \tag{7.11}$$

The corresponding derivation process has been shown in Formula 7.12.

$$\begin{aligned} &\log p(Q|\theta_{\bar{D}}) \\ &= \sum_{w \in Q} c(w,Q) \log p(w|\theta_{\bar{D}}) \\ &= \sum_{w \in Q \cap D} c(w,Q) \log \left(\frac{\mu}{\delta|V|+\mu} p(w|C)\right) + \sum_{w \in Q, w \notin D} c(w,Q) \log \left(\frac{\delta}{\delta|V|+\mu} + \frac{\mu}{\delta|V|+\mu} p(w|C)\right) \\ &= \sum_{w \in Q \cap D} c(w,Q) \log \left(\frac{\mu p(w|C)}{\delta|V|+\mu}\right) + \underbrace{\sum_{w \in Q} c(w,Q) \log \left(\frac{\delta + \mu p(w|C)}{\delta|V|+\mu}\right)}_{\text{document independent constant}} - \sum_{w \in Q \cap D} c(w,Q) \log \left(\frac{\delta + \mu p(w|C)}{\delta|V|+\mu}\right) \\ &\stackrel{rank}{=} -\sum_{w \in Q \cap D} c(w,Q) \log \left(1 + \frac{\delta}{\mu p(w|C)}\right) \end{aligned} \tag{7.12}$$

Plugging Equations 2.8 and 7.11 into Equation 7.10, we finally obtain a more complete query likelihood retrieval function that also contains the negative query generation component:

$$O(R=\ell|Q,D) \stackrel{rank}{=} \sum_{w \in Q \cap D} c(w,Q) \left[\log \left(1 + \frac{c(w,D)}{\mu p(w|C)}\right) + \log \left(1 + \frac{\delta}{\mu p(w|C)}\right)\right] + |Q| \log \frac{\mu}{|D|+\mu} \tag{7.13}$$

Comparing Formula 7.13 with the standard query likelihood in Formula 2.8, we can see that our new retrieval function essentially introduces a novel component $\log \left(1 + \frac{\delta}{\mu p(w|C)}\right)$ to reward the matching of a query term, and it rewards more the matching of a more discriminative query term, which not only intuitively makes sense, but also provides a natural way to incorporate IDF weighting to query likelihood, which has so far only been possible through a second-stage smoothing step [42, 122]. Note that when we set $\delta = 0$, the proposed retrieval function degenerates to the standard query likelihood function.

Furthermore, since this new component we introduced is a *term-dependent constant*, the proposed new retrieval function only incurs $\mathcal{O}(|Q|)$ additional computation cost as compared to the standard query likelihood function, which can be certainly ignored.

More interestingly, the developed query likelihood with negative query generation (Formula 7.13) leads to the same ranking formula as derived by lower-bounding term frequency normalization in the standard query likelihood method in Chapter 6. However, the Formula 6.14 derived in Chapter 6 is purely based on a heuristic approach, which is inconsistent with the probabilistic framework of the query likelihood method. Our method essentially provides a probabilistic approach for appropriately lower-bounding term frequency normalization in the standard query likelihood method.

7.4 A General Probabilistic Distance Retrieval Method

Query language models play a critical role in the language modeling approaches. However, similar to the standard query likelihood, a major deficiency of the proposed query likelihood with negative query generation is that it cannot easily incorporate query language models [120]. To address this problem, we further develop a more general probabilistic distance retrieval method to explicitly incorporate the query language model, inspired by the development of the KL-divergence retrieval method [58].

Specifically, given the regular document language model θ_D, the regular query language model θ_Q, and the proposed negative document language model $\theta_{\bar{D}}$ (Equation 7.9), we score a document D with respect to a query Q based on the difference of two KL-divergence values: one is the KL-divergence between the query language model θ_Q and the regular document language model θ_D, and the other KL-divergence between θ_Q and the negative document language model $\theta_{\bar{D}}$. Formally,

$$\begin{aligned} Score(D,Q) &= D(\theta_Q||\theta_{\bar{D}}) - D(\theta_Q||\theta_D) \\ &= \sum_{w\in V} p(w|\theta_Q)\log\frac{p(w|\theta_Q)}{p(w|\theta_{\bar{D}})} - \sum_{w\in V} p(w|\theta_Q)\log\frac{p(w|\theta_Q)}{p(w|\theta_D)} \\ &= \sum_{w\in V} p(w|\theta_Q)\log\frac{p(w|\theta_D)}{p(w|\theta_{\bar{D}})} \end{aligned} \tag{7.14}$$

This way, the closer the document language model is to the query language model and the farther the negative document language model is away from the query language model, the higher the document would be ranked.

Moreover, it is easy to show that this probabilistic distance retrieval method covers the query likelihood with

negative query generation as its special case when we use the empirical query word distribution to estimate θ_Q, i.e.,

$$p(w|\theta_Q) = \frac{c(w,Q)}{|Q|} \tag{7.15}$$

Indeed, with such an estimate, we have :

$$\begin{aligned} Score(D,Q) &= \sum_{w \in V} \frac{c(w,Q)}{|Q|} \log \frac{p(w|\theta_D)}{p(w|\theta_{\bar{D}})} \\ &\stackrel{rank}{=} \sum_{w \in V} c(w,Q) \log \frac{p(w|\theta_D)}{p(w|\theta_{\bar{D}})} \\ &= \log p(Q|\theta_D) - \log p(Q|\theta_{\bar{D}}) \end{aligned} \tag{7.16}$$

In this sense, the proposed probabilistic distance retrieval method is not only an extended KL-divergence model with a negative document language model but also a generalization of the proposed query likelihood scoring method with a negative query generation component, suggesting that the proposed probabilistic distance retrieval method would work more effectively than the basic KL-divergence retrieval method.

7.5 Summary

In this chapter, we show that we can improve the standard query likelihood function by bringing back the component of negative query generation (i.e., the probability that a user who dislikes a document would use a query). We argue that ignoring the component of negative query generation to justify the standard query likelihood retrieval function in existing work is questionable, because in reality, a user who dislikes a document would more likely avoid using words in the document when posing a query. We propose an effective approach to estimate document-dependent probabilities of negative query generation based on the principle of maximum entropy, and derive a more complete query likelihood retrieval function that contains the negative query generation component, which essentially scores a document with respect to a query according to the ratio of the probability that a user who likes the document would pose the query to the probability that a user who dislikes the document would pose the query. In addition, we further develop a more general probabilistic distance retrieval method to naturally incorporate query language models, which covers the proposed query likelihood with negative query generation as its special case.

Our work not only bridges the theoretic gap between the standard query likelihood method and the probability ranking principle, but also improves retrieval effectiveness over the standard query likelihood with no additional computational cost for various types of queries across different collections, as shown in Chapter 6. The proposed retrieval functions can potentially replace the standard query likelihood retrieval method and the standard KL-divergence re-

trieval method in all retrieval applications.

As a first attempt at bringing back negative query generation, our work opens up an interesting novel research direction in optimizing language models for information retrieval through improving the estimation of negative document language model. One of the most interesting directions is to study whether setting a term-specific δ can further improve performance. For example, term necessity prediction [124] and discovery of key concepts [7] could be two possible ways for setting adaptive δ. Another interesting direction is to go beyond document D and seek other resources for estimating a more accurate negative query generation probability.

Chapter 8

Conclusions and Future Work

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

8.1 Conclusions

Improving the effectiveness of general retrieval models has been a long-standing difficult challenge in information retrieval research, yet is also a fundamentally important task, because an improved general retrieval model would benefit every search engine. The language modeling framework for information retrieval have recently attracted much attention, due to its potential to develop an optimal retrieval model that is both theoretically sound and able to perform well empirically. However, the difficulty in accurately modeling the highly empirical notion of relevance within a standard statistical language model has led to slow progress in optimizing language modeling approaches; after more than one decade of research, the basic language modeling approach to retrieval still remains the same. This suggests that the theoretical framework of language models, without being able to accurately model the empirical notion of relevance, has a clear gap from what is needed to make a retrieval model empirically effective, a general problem we refer to as the "theory-effectiveness gap".

This thesis presents our efforts at bridging the "theory-effectiveness gap" between the theoretical framework of standard language models and the empirical application of information retrieval. Specifically, we clearly identified the causes of these gaps, and developed a series of general methodologies to remove the causes from language models without destroying their statistical foundation to improve language models from different perspectives, corresponding to the three main components of language modeling approaches, i.e., document language models, query language models, and the query likelihood scoring function. Specifically, this thesis makes contributions in the following five aspects.

- **Positional document language models:** we propose a novel positional language model (PLM) which implements term position and proximity heuristics in a unified language model. The key idea is to define a language model for each position of a document, and score a document based on the scores of its positional language models. We estimate PLM using a novel smoothing strategy based on different proximity-based kernel func-

tions. In addition, we work out a mathematical approach to reduce the computational complexity dramatically. The PLM has shown more robust and effective than the standard document language model, the state-of-the-art passage retrieval, and a state-of-the-art proximity retrieval model.

- **Positional relevance models for estimating query language models**: we propose a novel positional relevance model (PRM) for estimating query language models based on pseudo-relevance feedback. The PRM exploits term position and proximity evidence to assign more weights to words closer to query words based on the intuition that words closer to query words are more likely to be consistent with the query topic. An important advantage of the this method is that it can model the "relevant positions" in a feedback document with probabilistic models so as to assign more weights to terms at more relevant positions in a principled way, thus leading naturally to selection of expansion terms more likely relevant to the query topic. Besides, we develop two methods to estimate the proposed positional relevance model based on different sampling processes. Experiment results show that the proposed method is more effective and robust than current state-of-the-art pseudo-relevance feedback approaches.

- **Boosting the robustness of pseudo-relevance feedback approaches for estimating query language models**: we propose a novel learning algorithm based on the boosting framework to optimize pseudo-relevance feedback approaches for estimating query language models. A major contribution of our work is to optimize pseudo-relevance feedback based on a novel loss function that directly measures both robustness and effectiveness, which has not been achieved in any previous work. The experiment results demonstrate that our algorithm can achieve better average precision and meanwhile dramatically reduce the number and magnitude of feedback failure cases. Moreover, the proposed algorithm is actually more general and applicable to pseudo-relevance feedback in other retrieval models as well.

- **Lower-bounding the query likelihood scoring function**: we reveal a previously unknown deficiency of the query likelihood scoring function: it is not properly lower-bounded for long documents. As a result of this deficiency, long documents which do match the query term can often be scored unfairly as having a lower relevancy than shorter documents that do not contain the query term at all. In order to analytically diagnose this problem, we propose two desirable formal constraints to capture the heuristic of lower-bounding term frequency normalization, and use constraint analysis to examine the query likelihood function. Analysis results show that it can only satisfy the constraints for a certain range of parameter values and/or for a particular set of query terms. Empirical results further show that the retrieval performance tends to be poor when the parameter is out of the range or the query term is not in the particular set. To solve this problem, we propose an efficient method to introduce a sufficiently large lower bound for term frequency normalization which can be shown analytically to

fix the problem. Our experimental results demonstrate that the proposed method, incurring almost no additional computational cost, can improve the retrieval performance of the query likelihood function significantly. In fact, our empirical observation and constraint analysis show that the problem of improper lower-bound of term frequency normalization is a common deficiency of current retrieval models, including BM25, the pivoted length normalization method, etc. And the proposed method is a general solution that can be used as a plug-and-play patch to multiple state-of-the-art retrieval models to improve their effectiveness.

- **Query likelihood with negative query generation:** we improve the basic query likelihood function by bringing back the component of negative query generation (i.e., the probability that a user who dislikes a document would use a query) that was ignored. We argue that ignoring the component of negative query generation to justify the standard query likelihood retrieval function in existing work is questionable, because in reality, a user who dislikes a document would more likely avoid using words in the document when posing a query. We propose an effective approach to estimate document-dependent probabilities of negative query generation based on the principle of maximum entropy, and derive a more complete query likelihood retrieval function that contains the negative query generation component, which essentially scores a document with respect to a query according to the ratio of the probability that a user who likes the document would pose the query to the probability that a user who dislikes the document would pose the query. In addition, we further develop a more general probabilistic distance retrieval method to naturally incorporate feedback, which covers the proposed query likelihood with negative query generation as its special case. The proposed approach not only bridges the theoretical gap between the standard query likelihood and the probability ranking principle, but also improves retrieval effectiveness over the standard query likelihood with no additional computational cost. More interestingly, the developed query likelihood with negative query generation leads to the same ranking formula as derived by lower-bounding the query likelihood scoring function, thus essentially providing a probabilistic interpretation for the heuristic way of lower-bounding term frequency normalization in the basic query likelihood method.

In summary, this thesis proposes to improve the effectiveness language modeling approaches to information retrieval through bridging the "theory-effectiveness gap", and has resulted in several more effective and robust retrieval algorithms. All the proposed new models are general, and thus can be used immediately in any search engine to improve its retrieval accuracy over the current retrieval models.

Although we focus on language models in this thesis, most of the proposed methodologies are actually not restricted to language models and can also be applied to retrieval models other language models.

8.2 Future Work

There are still many challenges to be solved to fully develop the potential of language models by bridging the theory-effectiveness gap. The following is a list of some interesting opportunities for future research.

A unified model to bridge all the identified theory-effective gaps: we have presented different algorithms for bridging multiple theory-effectiveness gaps of language modeling approaches from different perspectives. A direct future work is thus to evaluate if the benefits from different algorithms can add together; in other words, how can we develop a unified model that is able to bridge all these gaps in a single effective retrieval function? However, it is a non-trivial task, as it would raise some problems if we simply put these algorithms together, due to the potential overlap between different algorithms. For example, because positional language models and the lower-bounded query likelihood function (or query likelihood with negative query generation) both can improve the probability of retrieving long documents, though from bridging different gaps, we should pay careful attention to avoid overly-rewarding long documents if we use both algorithms together; positional language models and positional relevance models both consider the term proximity evidence, so applying them together may count the term proximity effect twice. The axiomatic approach, as we used to successfully develop the lower-bounded query likelihood retrieval function, could be a promising way to leverage the proposed algorithms together by formalizing retrieval constraints to regularize the overlap between different algorithms.

Adaptive language models: one deficiency of current language models is that document characteristics are not fully explored, leading to non-optimal retrieval performance. For example, the content of a news article is usually coherent and focused on one topic, while the content of a forum thread is generally incoherent and often covers multiple different topics. Unfortunately, current language models do not consider this difference in collection characteristics, thus facing a dilemma: if we use a whole document as a unit for estimating language models, it would be appropriate for news data, but it would not perform well for forum data; on the other hand, if we estimate language models based on its best-matching passage, which likely helps forum data, but it would be non-optimal for news data. Thus adapting language models to document characteristics is needed to improve search performance. The proposed positional document language model provides a framework to adapt document language models to document characteristics by dynamically choosing different kernel functions. It would be interesting to further explore advanced NLP techniques (e.g., discourse structure analysis) and machine learning approaches to develop an adaptive language model.

Effective language models for retrieval with complex information needs: language models are natural for modeling topical relevance. But in many retrieval applications, a user's information need consists of multiple dimensions of preferences with topical relevance being only one of them. Other factors such as readability, genre, reliability, and sentiment may also be important. How can we use language models to capture such non-topical aspects, and develop language models to optimize ranking of documents based on multiple factors are still unclear. In this direction,

recent work has shown that the learning-to-rank approaches [62] are quite promising, thus again it would be very interesting to study how to leverage the language modeling approaches (generative approaches) with the learning-to-rank approaches (discriminative approaches).

An ultimately optimal retrieval model: our goal is to develop a robust and effective language model for information retrieval that can (1) adapt retrieval parameters to different queries, documents, and tasks, (2) optimize retrieval parameters automatically, (3) perform as well as or better than well-tuned traditional retrieval methods, and (4) be computed as efficiently as traditional retrieval methods.

References

[1] Nasreen Abdul-jaleel, James Allan, W. Bruce Croft, O Diaz, Leah Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. Umass at trec 2004: Novelty and hard. In *Proceedings of TREC-13*, 2004.

[2] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 1964.

[3] James Allan. Relevance feedback with too much data. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '95, pages 337–343, New York, NY, USA, 1995. ACM.

[4] Giambattista Amati, Claudio Carpineto, and Giovanni Romano. Query difficulty, robustness, and selective application of query expansion. In *Proceedings of the 25th European Conference on Information Retrieval*, ECIR '04, pages 127–137, 2004.

[5] Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, October 2002.

[6] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. Formal models for expert finding in enterprise corpora. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 43–50, New York, NY, USA, 2006. ACM.

[7] Michael Bendersky and W. Bruce Croft. Discovering key concepts in verbose queries. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 491–498, New York, NY, USA, 2008. ACM.

[8] Adam Berger and John Lafferty. Information retrieval as statistical translation. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 222–229, New York, NY, USA, 1999. ACM.

[9] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998.

[10] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85, June 1990.

[11] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using smart: Trec 3. In *Proceedings of TREC-3*, pages 69–80, 1994.

[12] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[13] Georg Buscher, Andreas Dengel, and Ludger van Elst. Query expansion using gaze-based feedback on the subdocument level. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 387–394, New York, NY, USA, 2008. ACM.

[14] Stefan Büttcher and Charles L. A. Clarke. Efficiency vs. effectiveness in terabyte-scale information retrieval. In *Proceedings of TREC 2005*, 2005.

[15] Stefan Büttcher, Charles L. A. Clarke, and Brad Lushman. Term proximity scoring for ad-hoc retrieval on very large text collections. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 621–622, New York, NY, USA, 2006. ACM.

[16] James P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 302–310, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[17] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 243–250, New York, NY, USA, 2008. ACM.

[18] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM.

[19] Ben Carterette, James Allan, and Ramesh Sitaraman. Minimal test collections for retrieval evaluation. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 268–275, New York, NY, USA, 2006. ACM.

[20] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. Shortest substring ranking (multitext experiments for trec-4). In *Proceedings of TREC-4*, pages 295–304, 1995.

[21] Stéphane Clinchant and Eric Gaussier. Information-based models for ad hoc ir. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 234–241, New York, NY, USA, 2010. ACM.

[22] Kevyn Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 837–846, New York, NY, USA, 2009. ACM.

[23] Kevyn Collins-Thompson and Jamie Callan. Estimation and use of uncertainty in pseudo-relevance feedback. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 303–310, New York, NY, USA, 2007. ACM.

[24] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 299–306, New York, NY, USA, 2002. ACM.

[25] Ronan Cummins and Colm O'Riordan. An axiomatic comparison of learned term-weighting schemes in information retrieval: clarifications and extensions. *Artif. Intell. Rev.*, 28(1):51–68, June 2007.

[26] Owen de Kretser and Alistair Moffat. Effective document presentation with a locality-based similarity heuristic. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 113–120, New York, NY, USA, 1999. ACM.

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[28] Fernando Diaz and Donald Metzler. Improving the estimation of relevance models using large external corpora. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 154–161, New York, NY, USA, 2006. ACM.

[29] Hui Fang, Tao Tao, and ChengXiang Zhai. A formal study of information retrieval heuristics. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 49–56, New York, NY, USA, 2004. ACM.

[30] Hui Fang, Tao Tao, and Chengxiang Zhai. Diagnostic evaluation of information retrieval models. *ACM Trans. Inf. Syst.*, 29:7:1–7:42, April 2011.

[31] Hui Fang and ChengXiang Zhai. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 480–487, New York, NY, USA, 2005. ACM.

[32] Hui Fang and ChengXiang Zhai. Semantic term matching in axiomatic approaches to information retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 115–122, New York, NY, USA, 2006. ACM.

[33] Hui Fang and ChengXiang Zhai. Probabilistic models for expert finding. In *Proceedings of the 29th European conference on IR research*, ECIR'07, pages 418–430, Berlin, Heidelberg, 2007. Springer-Verlag.

[34] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.

[35] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, pages 23–37, London, UK, UK, 1995. Springer-Verlag.

[36] Norbert Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35:243–255, 1992.

[37] Jianfeng Gao, Jian-Yun Nie, Guangyuan Wu, and Guihong Cao. Dependence language model for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 170–177, New York, NY, USA, 2004. ACM.

[38] Donna Harman and Chris Buckley. The nrrc reliable information access (ria) workshop. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 528–529, New York, NY, USA, 2004. ACM.

[39] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[40] David Hawking and Paul B. Thistlewaite. Proximity operators - so near and yet so far. In *TREC '95*, pages 500–236, 1995.

[41] Ben He and Iadh Ounis. Finding good feedback documents. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 2011–2014, New York, NY, USA, 2009. ACM.

[42] Djoerd Hiemstra. A probabilistic justification for using tf.idf term weighting in information retrieval. *International Journal on Digital Libraries*, 3(2):131–139, 2000.

[43] Djoerd Hiemstra. *Using language models for information retrieval*. PhD thesis, University of Twente, 2001.

[44] Djoerd Hiemstra, Stephen Robertson, and Hugo Zaragoza. Parsimonious language models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 178–185, New York, NY, USA, 2004. ACM.

[45] E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106(4):620–630, May 1957.

[46] Frederick Jelinek. *Statistical methods for speech recognition*. MIT Press, Cambridge, MA, USA, 1997.

[47] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[48] Karen S. Jones and Steven E. Robertson. LM vs PM: Where's the Relevance? In Jamie Callan, Bruce W. Croft, and John Lafferty, editors, *Proceedings of the Workshop on Language Modeling and Information Retrieval*, pages 12–15, Pittsburgh, PA, 2001. Carnegie Mellon University.

[49] Karen Sparck Jones, Stephen E. Robertson, Djoerd Hiemstra, and Hugo Zaragoza. Language modeling and relevance. In Bruce W. Croft and John Lafferty, editors, *Language Modeling for Information Retrieval*, volume 13 of *The Kluwer International Series on Information Retrieval*. Kluwer Academic Publishers, 2003.

[50] Karen Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Inf. Process. Manage.*, 36:779–808, November 2000.

[51] Marcin Kaszkiel and Justin Zobel. Passage retrieval revisited. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '97, pages 178–185, New York, NY, USA, 1997. ACM.

[52] Marcin Kaszkiel and Justin Zobel. Effective ranking with arbitrary passages. *Journal of the American Society for Information Science and Technology*, 52(4):344–364, 2001.

[53] Marcin Kaszkiel, Justin Zobel, and Ron Sacks-Davis. Efficient passage ranking for document databases. *ACM Transactions on Information Systems*, 17(4):406–439, 1999.

[54] E. Michael Keen. The use of term position devices in ranked output experiments. *The Journal of Documentation*, 47(1):1–22, 1991.

[55] E. Michael Keen. Some aspects of proximity searching in text retrieval systems. *Journal of Information Science*, 18(2):89–98, 1992.

[56] Koichi Kise, Markus Junker, Andreas Dengel, and Keinosuke Matsumoto. *Passage Retrieval Based on Density Distributions of Terms and Its Applications to Document Retrieval and Question Answering*, volume 2956 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 2004.

[57] Oren Kurland, Lillian Lee, and Carmel Domshlak. Better than the real thing?: iterative pseudo-query processing using cluster-based language models. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 19–26, New York, NY, USA, 2005. ACM.

[58] John Lafferty and Chengxiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 111–119, New York, NY, USA, 2001. ACM.

[59] John Lafferty and Chengxiang Zhai. Probabilistic relevance models based on document and query generation. In *Language Modeling and Information Retrieval*, pages 1–10. Kluwer Academic Publishers, 2002.

[60] Victor Lavrenko, Martin Choquette, and W. Bruce Croft. Cross-lingual relevance models. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 175–182, New York, NY, USA, 2002. ACM.

[61] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 120–127, New York, NY, USA, 2001. ACM.

[62] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[63] Xiaoyong Liu and W. Bruce Croft. Passage retrieval based on language models. In *Proceedings of the eleventh international conference on Information and knowledge management*, CIKM '02, pages 375–382, New York, NY, USA, 2002. ACM.

[64] Xiaoyong Liu and W Bruce Croft. Statistical language modeling for information retrieval. *Annual Review of Information Science and Technology*, 39, 2003.

[65] Xiaoyong Liu and W. Bruce Croft. Cluster-based retrieval using language models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 186–193, New York, NY, USA, 2004. ACM.

[66] David E. Losada and Leif Azzopardi. An analysis on document length retrieval trends in language modeling smoothing. *Inf. Retr.*, 11:109–138, April 2008.

[67] Hans P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM J. Res. Dev.*, 1:309–317, October 1957.

[68] Yuanhua Lv and ChengXiang Zhai. Adaptive relevance feedback in information retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 255–264, New York, NY, USA, 2009. ACM.

[69] Yuanhua Lv and ChengXiang Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1895–1898, New York, NY, USA, 2009. ACM.

[70] Yuanhua Lv and ChengXiang Zhai. Adaptive term frequency normalization for bm25. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 1985–1988, New York, NY, USA, 2011. ACM.

[71] Yuanhua Lv and ChengXiang Zhai. When documents are very long, bm25 fails! In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 1103–1104, New York, NY, USA, 2011. ACM.

[72] Qiaozhu Mei, Hui Fang, and ChengXiang Zhai. A study of poisson query generation model for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 319–326, New York, NY, USA, 2007. ACM.

[73] Annabelle Mercier and Michel Beigbeder. Fuzzy proximity ranking with boolean queries. In *Proceedings of TREC '05*, 2005.

[74] Donald Metzler and W. Bruce Croft. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 472–479, New York, NY, USA, 2005. ACM.

[75] Donald Metzler and W. Bruce Croft. Latent concept expansion using markov random fields. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 311–318, New York, NY, USA, 2007. ACM.

[76] Donald Metzler, Victor Lavrenko, and W. Bruce Croft. Formal multiple-bernoulli models for language modeling. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 540–541, New York, NY, USA, 2004. ACM.

[77] David R. H. Miller, Tim Leek, and Richard M. Schwartz. A hidden markov model information retrieval system. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 214–221, New York, NY, USA, 1999. ACM.

[78] Christof Monz. Minimal span weighting retrieval for question answering. In Rob Gaizauskas, Mark Greenwood, and Mark Hepple, editors, *SIGIR Workshop on Information Retrieval for Question Answering*, pages 23–30, 2004.

[79] Paul Ogilvie and Jamie Callan. Combining document representations for known-item search. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 143–150, New York, NY, USA, 2003. ACM.

[80] Desislava Petkova and W. Bruce Croft. Proximity-based document representation for named entity retrieval. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 731–740, New York, NY, USA, 2007. ACM.

[81] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 275–281, New York, NY, USA, 1998. ACM.

[82] Yves Rasolofo and Jacques Savoy. Term proximity scoring for keyword-based retrieval systems. In *Proceedings of the 25th European Conference on IR Research (ECIR 2003)*, pages 207–218, 2003.

[83] Stephen Robertson and Djoerd Hiemstra. Language models and probability of relevance. In Jamie Callan, Bruce W. Croft, and John Lafferty, editors, *In Proceedings of the first Workshop on Language Modeling and Information Retrieval*, pages 21–25, Pittsburgh, PA, 2001. Carnegie Mellon University.

[84] Stephen E. Robertson. The probability ranking principle in ir. *Journal of Documentation*, 33(4):294–304, 1977.

[85] Stephen E. Robertson and Karen Sparck Jones. Relevance weighting of search terms. *Journal of the American Society of Information Science*, 27(3):129–146, 1976.

[86] Stephen E. Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 232–241, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[87] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. In *Proceedings of TREC-3*, pages 109–126, 1994.

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

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

[90] Gerard Salton, J. Allan, and Chris Buckley. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, pages 49–58, New York, NY, USA, 1993. ACM.

[91] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society of Information Science*, 41(4):288–297, 1990.

[92] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the eleventh annual conference on Computational learning theory*, COLT' 98, pages 80–91, New York, NY, USA, 1998. ACM.

[93] C. E. Shannon. Prediction and entropy of printed english. *Bell Sys. Tech. Jour.*, pages 51–64, 1951.

[94] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 43–50, New York, NY, USA, 2005. ACM.

[95] Luo Si, Rong Jin, Jamie Callan, and Paul Ogilvie. A language modeling framework for resource selection and results merging. In *Proceedings of the eleventh international conference on Information and knowledge management*, CIKM '02, pages 391–397, New York, NY, USA, 2002. ACM.

[96] Amit Singhal. Modern information retrieval: a brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24:2001, 2001.

[97] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '96, pages 21–29, New York, NY, USA, 1996. ACM.

[98] Fei Song and W. Bruce Croft. A general language model for information retrieval. In *Proceedings of the eighth international conference on Information and knowledge management*, CIKM '99, pages 316–321, New York, NY, USA, 1999. ACM.

[99] Ruihua Song, Michael J. Taylor, Ji-Rong Wen, Hsiao-Wuen Hon, and Yong Yu. Viewing term proximity from a different perspective. In *Proceedings of the IR research, 30th European conference on Advances in information retrieval*, ECIR'08, pages 346–357, Berlin, Heidelberg, 2008. Springer-Verlag.

[100] Natali Soskin, Oren Kurland, and Carmel Domshlak. Navigating in the dark: Modeling uncertainty in ad hoc retrieval using multiple relevance models. In *Proceedings of the 2nd International Conference on Theory of Information Retrieval: Advances in Information Retrieval Theory*, ICTIR '09, pages 79–91, Berlin, Heidelberg, 2009. Springer-Verlag.

[101] Karen Sparck Jones and Peter Willett, editors. *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[102] Bin Tan, Xuehua Shen, and ChengXiang Zhai. Mining long-term search history to improve search accuracy. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 718–723, New York, NY, USA, 2006. ACM.

[103] Tao Tao and ChengXiang Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 162–169, New York, NY, USA, 2006. ACM.

[104] Tao Tao and ChengXiang Zhai. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 295–302, New York, NY, USA, 2007. ACM.

[105] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 41–47, New York, NY, USA, 2003. ACM.

[106] Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. Hypergeometric language models for republished article finding. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 485–494, New York, NY, USA, 2011. ACM.

[107] C. J. van Rijsbergen. A non-classical logic for information retrieval. *Comput. J.*, 29(6):481–485, 1986.

[108] Olga Vechtomova and Ying Wang. A study of the effect of term proximity on query expansion. *Journal of Information Science*, 32(4):324–333, August 2006.

[109] Xuanhui Wang, Hui Fang, and ChengXiang Zhai. A study of methods for negative relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 219–226, New York, NY, USA, 2008. ACM.

[110] Xing Wei and W. Bruce Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 178–185, New York, NY, USA, 2006. ACM.

[111] 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.

[112] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *SIGIR '96*, pages 4–11, 1996.

[113] Jinxi Xu and W. Bruce Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 254–261, New York, NY, USA, 1999. ACM.

[114] Jinxi Xu, Ralph Weischedel, and Chanh Nguyen. Evaluating a probabilistic model for cross-lingual information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 105–110, New York, NY, USA, 2001. ACM.

[115] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 391–398, New York, NY, USA, 2007. ACM.

[116] Shipeng Yu, Deng Cai, Ji-Rong Wen, and Wei-Ying Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 11–18, New York, NY, USA, 2003. ACM.

[117] Hugo Zaragoza, Djoerd Hiemstra, and Michael Tipping. Bayesian extension to the language model for ad hoc information retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 4–9, New York, NY, USA, 2003. ACM.

[118] ChengXiang Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2:137–213, March 2008.

[119] ChengXiang Zhai, William W. Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 10–17, New York, NY, USA, 2003. ACM.

[120] Chengxiang Zhai and John Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 403–410, New York, NY, USA, 2001. ACM.

[121] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 334–342, New York, NY, USA, 2001. ACM.

[122] ChengXiang Zhai and John Lafferty. Two-stage language models for information retrieval. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 49–56, New York, NY, USA, 2002. ACM.

[123] Yi Zhang, Jamie Callan, and Thomas Minka. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 81–88, New York, NY, USA, 2002. ACM.

[124] Le Zhao and Jamie Callan. Term necessity prediction. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 259–268, New York, NY, USA, 2010. ACM.

[125] Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 287–294, New York, NY, USA, 2007. ACM.