# MAKING A BETTER QUERY: FIND GOOD FEEDBACK DOCUMENTS AND TERMS VIA SEMANTIC ASSOCIATIONS

JUN MIAO

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO
JUNE, 2016

# Abstract

When people search, they always input several keywords as an input query. While current information retrieval (IR) systems are based on term matching, documents will not be considered as relevant if they do not have the exact terms as in the query. However, it is common that these documents are relevant if they contain terms semantically similar to the query. To retrieve these documents, a classic way is to expand the original query with more related terms. Pseudo relevance feedback (PRF) has proven to be effective in expanding origin queries and improve the performance of IR. It assumes the top "k" ranked documents obtained through the first round retrieval are relevant as "feedback documents", and expand the original queries with "feedback terms" selected from these feedback documents.

However, applying PRF for query expansion must be very carefully. Wrongly added terms can bring noisy information and hurt the overall search experiences extensively. The assumption of "feedback documents" is too strong in real cases. To avoid noise import and make significant improvements simultaneously, we solve the significant problem through four ways in this dissertation. Firstly, we assume the proximity information among terms as term semantic associations and utilize them to seek new relevant terms. Next, to obtain good and robust performance for PRF via adapting topic information, we propose a new concept named “topic space” and present three models based on it. Topics obtained through topic modeling do help identify how relevant a feedback document is. Weights of candidate terms in these more relevant feedback documents will be boosted and have higher probabilities to be chosen. Furthermore, we apply machine learning methods to classify which feedback documents are effective for PRF. To solve the problem of lack-of-training-data for the application of machine learning methods in PRF, we improve a traditional co-training method and take the quality of classifiers into account. Finally, we present a new probabilistic framework to integrate existing effective methods like semantic associations as components for further research. All the work has been tested on public datasets and proven to be effective and efficient.

# Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Jimmy Huang for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Besides my supervisor, I would like to thank the rest of my thesis committee: Prof.John Tsotsos, Prof. Jianghong Wu, Prof. Aijun An, Prof. Jamie Callan and Prof.George Georgopoulos, for their insightful comments, encouragement and hard questions which drive me to widen my research from various perspectives. Prof. Nick Cercone, who was my previous thesis committee, was gone during the period of my defence preparation. I really appreciated what he had done for me and wish him R.I.P.

I thank my fellow labmates: Jiashu Zhao, Zheng Ye, Xiaofeng Zhou, Ben He, Qinmin Hu and so on, for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last seven years.

Last but not least, I would like to thank my family. My beloved father Xuyi Miao and mother Duanrong Tao always support me in my whole life. My wife Meimei Bai always encourages my study and research, takes care of me and calms me down when I have obstacles in my research. Because of them, I can pass the most difficult time and finish my thesis.

# Table of Contents

## List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background Knowledge of IR

People are hunger for tremendous information in their everyday lives. Beyond the knowledge scope of a person, there are lots of things he/she want to know, from the boyfriend of Jennifer Lawrence to the concept of graphic modeling. The search behavior can happen in lots of scenarios, e.g. web surfing, book searching or job hunting. That is why search technologies are indispensable for human beings in nowadays.

In the past, the primary challenge for searching was the lack of enough information. In the age of Internet, however, finding relevant information is now more important than just obtaining information. Wrong information can contribute to loss of time and money even lives! For example, the initial release of Apple map could navigate users to a train station in the middle of the sea [1].

In order to obtain particular information for their needs, people use Information

[1] `http://www.techradar.com/news/software/applications/ios-6-maps-what-went-wrong-1118121`

Retrieval (IR) technologies [RWJ+94a, Zha08a, ZHY14, ZHH+12, YHLZ13]. IR is actually a very general concept. Asking for the direction to a destination when lost is also a behavior of IR, so is looking for a book through cards in the libraries. In this thesis, the IR technologies we discuss are limited to digital text search systems such as search engines or domain-based search systems.

Recently, with the development of the Internet, people search more and more in a massive scale of resources stored in the different formats of media. In 2013, there were totally 2,161,530,000,000 Google searches, increasing more than 10% over the number 1,873,910,000,000 in 2012[2]. The tremendous numbers indicate that people's demands for information grow at an incredible speed. The increasing demands of users require better search results through such big data in a very short period, which bring plenty of challenges for researchers.

## 1.2 Problems and Motivation

Although music and figure searches are now available on the Internet, most search queries are still in the text format. People search what they want by inputting queries, which contain a few keywords that can describe their search intents. These queries, however, cannot accurately express what the users sometimes want and

[2]`http://www.statisticbrain.com/google-searches`

therefore prevent them from obtaining the real useful information. We consider this kind of queries to be of *low quality*. Low-quality queries can lead to some severe problems when proceeding bag-of-words matchings, which are implemented in most IR systems. Bag-of-words matchings suppose keywords in a query are independent and proceed the query-document matching word by word. The idea of bag-of-word makes IR or other applications like spam filtering or text classification straightforward and useful. Many current IR systems for businesses, libraries or local networks still rely on bag-of-words technologies [3].

Because the assumption of bag-of-words matching is simple but not right in the real-world, it has some deficiencies for conducting a real effective search in some cases, especially when the quality of user queries is low. Before demonstrating the ideas of our research work, we would like to explain the main reasons for the query quality problem with some examples. A typical scenario is that users are not clear about what they want to search. Sometimes, people only know some segments of the desired information. For example, when a user intends to search the story "Snow White" and does not know the exact name of the story, he or she may seek "fairytale princess witch queen" or something like that. However, there are many stories about princesses, witches, and queens. Documents which contain these keywords can be

[3]For example, York University library http://www.library.yorku.ca/web/ or Bestbuy/ Sear websites whose search function is implemented by Solr http://lucene.apache.org/solr/. Indices created by Solr are word-based

irrelevant. As a result, some or even most of the returned documents will be disappointing. In this case, it is important to detect the specific intents and narrow the search scope by extra information, e.g., contexts and user click through data.

Another kind of problems is caused by synonymy (different words with the same meaning), polysemy (one word with different meanings) and hyponymy (one word includes other words on the semantic level). Synonymy can result in a failure to retrieve relevant documents in a bag-of-words search. For example, if a user searches "mobile phone", he or she will miss the documents which only contain "cellphone" although the two terms mean the same thing. This search strategy decreases the recall rate (the fraction of the documents that are relevant to the query that is successfully retrieved) since these "cellphone"r ecords are very likely to be ignored. Meanwhile, the polysemy problem happens when a user searches "BMO bank". The demanded information should be about financial institutions. But because of the assumption of bag-of-words search, documents about river sides will be returned because they contain lots of "bank". For the search system, any documents include the keyword "bank" can be relevant too, no matter what their contents are about institutions or river sides. In such searches, the relation of "BMO" and "bank" can be informative but ignored. Part of the results is irrelevant. Thus, this decreases the precision rate (the fraction of retrieved documents that are relevant to the search) of the final results. Hyponymy problem frequently happens, especially when a domain-based search is conducted. People may want to explore "heart diseases" in a medical dataset. However, plenty of heart diseases are not shown as "heart diseases" in documents. They appear in specific names like "Endocarditis"or Cardiomyopathies". Documents with these words can be treated as irrelevant because they do not contain "heart diseases" literally. Although these documents are possible to be relevant, they will be ignored in a bag-of-words search.

Finally, sometimes users are eager to see more similar results according to their queries because they want to know something beyond their scope. While they search "Asus laptop" on the Internet, they are likely to see comparisons and reviews for other brands as well. As a result, a list of well-matched documents is not enough for their needs. We can describe the phenomenon as "desire of high recall rate".

It is impossible to require all users to write high-quality queries. So refining these queries automatically is hugely demanded. To deal with these problems, *query expansion* (QE) methods are introduced. Original queries are improved through different methods [ABA+01, CR12]. QE can be implemented mainly through interactive ways, relevance feedback (RF), word sense disambiguation (WSD), search result clustering or other techniques [CR12]. Interactive techniques show some candidates terms and require users to choose the best expansion terms from them. Relevance feedback uses the initial results obtained through some traditional IR methods and takes relevant documents selected by users and then selects some terms in these documents to refine the original queries. WSD tries to identify the real meaning of terms in a query through corpus analysis. Search result clustering summarizes a group of searched documents and discovers appropriate labels/terms to represent them. These labels/terms can be used to refine the original query. The core idea of QE is **mining terms which have particular associations to the users' origin query and expanding it with them.**

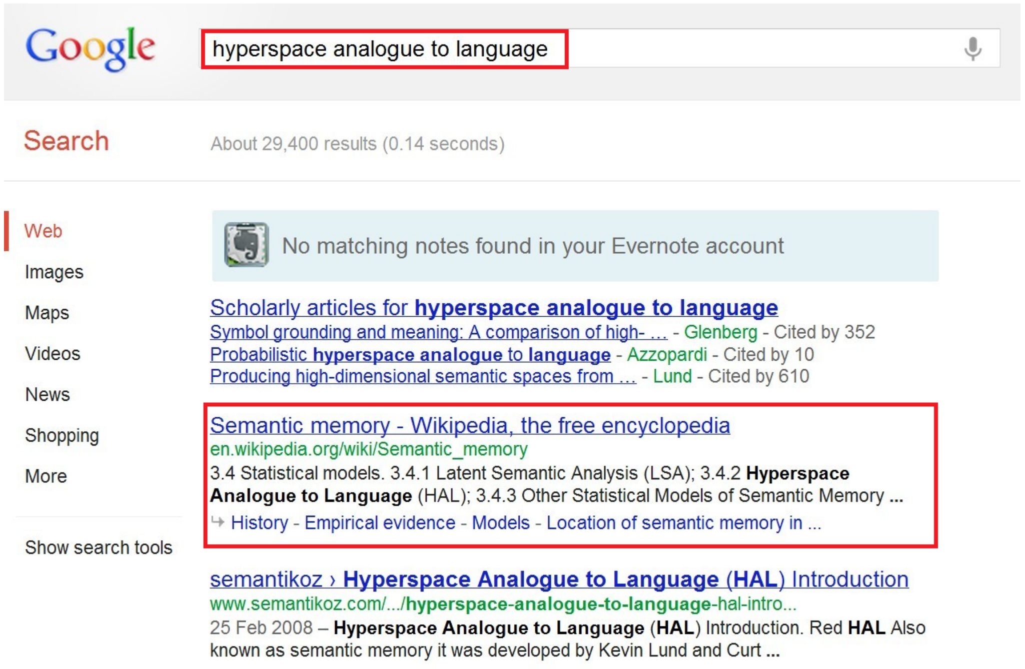
In the thesis, we mainly focus on how to utilize relevance feedback, to be more specific, pseudo relevance feedback (PRF) [MHZ16, MHH10, YHM12a] to reformulate queries for better results. PRF has proven to be effective for the synonyms-like problem and the "desire of high recall rate" phenomenon.

Initially, we would like to introduce the background knowledge of relevance feedback. Relevance feedback improves the query representation by taking feedback information into account. A traditional relevance feedback algorithm was proposed in [Roc71a] for the SMART retrieval system [Sal71]. It takes a set of documents as the feedback set. Unique terms in this set are ranked in a descending order of $tf \cdot idf$ weights [Ram03]. Some top-ranked feedback terms are then added to the original query, and finally documents are returned for the expanded query.

The feedback documents can be obtained by many possible means. In general, some methods utilize explicit evidence, such as labeled relevant documents from real users, while others use implicit evidence, such as the click-through data. Obtaining feedback information involves extra efforts, e.g. relevance judgment by real users, which is usually expensive. When conducting a search, users are generally unwilling to do extra work like annotating a document to be relevant or not. Meanwhile, for every given query, the corresponding feedback information is not necessarily available. There are not enough user data for a particular query especially for small information retrieval systems like library systems. An alternate solution is *pseudo-relevance feedback* (PRF) [Buc94, Eft96, LBCC04, MSB98, XC96, XC00]. PRF is an effective technology for QE. The basic idea of QE via PRF is selecting feedback terms from the top-ranked return documents by applying the original query (so-called first-pass retrieval), and then formulate a new query for the second round retrieval. As we can see, PRF uses feedback terms which appear together with query terms in the feedback documents. So we can have a similar summarization for QE via PRF like QE: the idea of PRF is **utilizing the co-occurrences of terms and query terms in a particular term space, i.e. feedback documents, as a term association to select related feedback terms and expand the origin query**. Through this kind of methods, some relevant documents missed in the initial round can then be retrieved to improve the overall performance. As we mentioned above, the research of QE via PRF is crucial and useful while user data are not applicable for many IR systems. Notably, PRF refers to QE based on PRF in the rest of this thesis.

Despite the marked improvement over the initial retrieval performance [Ama03a, RWHB+95], traditional PRF has its limits. There have been many studies on PRF's effectiveness. For example, a wide range of predictors were proposed to indicate the query performance, which is usually correlated with PRF's effectiveness [ACR04, CRG02, HO09b]. All these previous studies have agreed on a conclusion that the quality of feedback documents is crucial to PRF's performance. Since feedback documents are not assessed by real users when using PRF, the quality of the feedback document set is not guaranteed. In particular, a feedback document may not be useful even if it is relevant to the query topic. The document could be just partially relevant, and there might be only a small part of the document that is about the query topic, while the rest of the document is irrelevant. In this case, off-topic expansion terms are added to the query, leading to degraded retrieval performance. In other words, the term occurrence association is not a reliable choice for feedback term selection when the term space is not good. So *Relevance* is not enough to judge whether the document can help PRF to improve the performance in such scenarios. Thus, we use "quality" instead of "relevance" to evaluate a feedback document. The “quality” concept here is different from how we consider a query as we mentioned previously. Particularly, a “good/positive” quality feedback document is not only completely relevant but also useful in improving the final performance of PRF. Hence, there is a crucial need for selecting good quality feedback documents [HO09b, YHL11a]. Also, how to define a “good” feedback document is not determined yet.

Besides, not all terms in a feedback document are related to the original query, even if the document itself is highly related. A typical example is shown in Figure 1.1 and Figure 1.2. If we search “Hyperspace Analogue to Language” on Google, the Wikipedia page of semantic memory is returned in the first place. However, “Hyperspace Analogue to Language” is only a small part of this page. If we take all the terms as candidates in this page into account and treat them equally, it is very likely to import noisy terms since most topics are not about “Hyperspace Analogue to Language”. Traditional PRF methods will fail in this case. It is partially because the query itself was ignored in the process of feedback term selection. To be more specific, the co-occurrence associations between candidate terms and the query terms in the feedback documents are coarse for traditional PRF models. We do need a better association to choose feedback terms when term space is not good. But how to utilize different associations among terms to discover more relevant feedback terms


hyperspace analogue to language

Search

About 29,400 results (0.14 seconds)

Web
Images
Maps
Videos
News
Shopping
More

Show search tools

No matching notes found in your Evernote account

Scholarly articles for **hyperspace analogue to language**
Symbol grounding and meaning: A comparison of high- ... - Glenberg - Cited by 352
Probabilistic **hyperspace analogue to language** - Azzopardi - Cited by 10
Producing high-dimensional semantic spaces from ... - Lund - Cited by 610

Semantic memory - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Semantic_memory
3.4 Statistical models. 3.4.1 Latent Semantic Analysis (LSA); 3.4.2 **Hyperspace Analogue to Language** (HAL); 3.4.3 Other Statistical Models of Semantic Memory ...
↪ History - Empirical evidence - Models - Location of semantic memory in ...

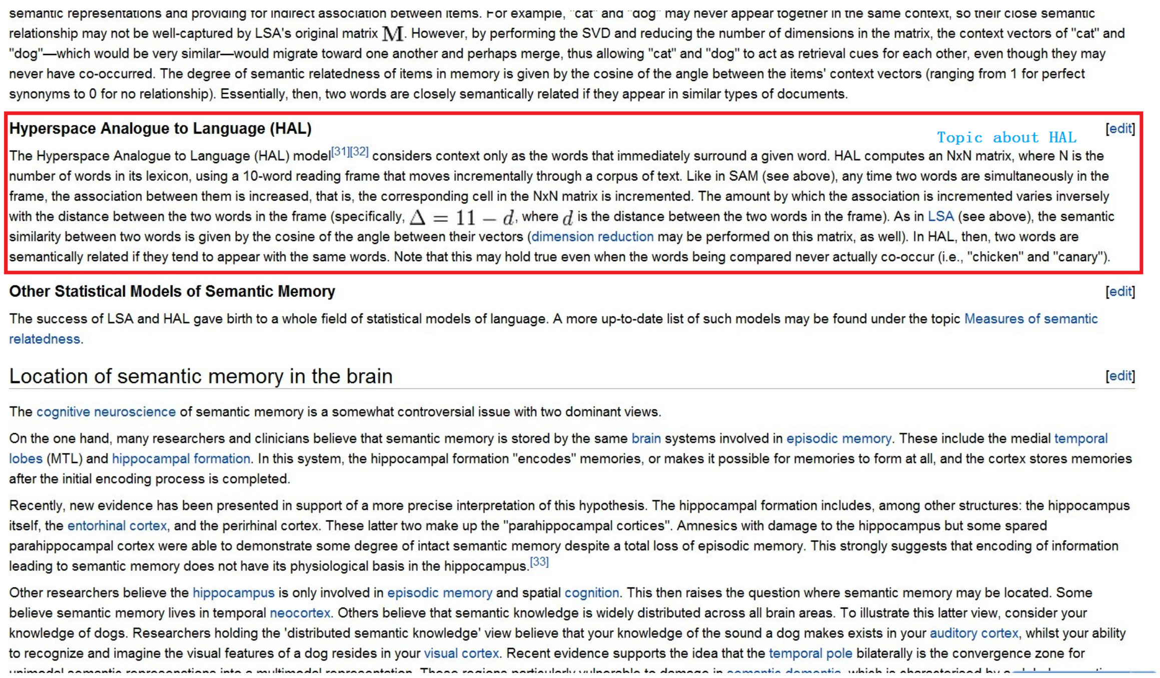
semantikoz › **Hyperspace Analogue to Language** (**HAL**) Introduction
www.semantikoz.com/.../**hyperspace-analogue-to-language**-hal-intro...
25 Feb 2008 – **Hyperspace Analogue to Language** (**HAL**) Introduction. Red **HAL** Also known as semantic memory it was developed by Kevin Lund and Curt ...

Figure 1.1: Search "Hyperspace Analogue to Language" on Google

is still under exploration.

## 1.3 Proposed Approaches

Intuitively, we consider discovering terms which are semantically related to the queries. Terms which have strong associations are supposed to be relevant and useful for PRF. Also, it is reasonable to prefer terms from feedback documents which are semantically related to the queries or similar to other relevant documents. "Semantically related" is a rough association which can be hardly identified by human beings. It is easy for people to consider "Shaquille O'Neal" is related to both "NBA" and


semantic representations and providing for indirect association between items. For example, "cat" and "dog" may never appear together in the same context, so their close semantic relationship may not be well-captured by LSA's original matrix $\mathbf{M}$. However, by performing the SVD and reducing the number of dimensions in the matrix, the context vectors of "cat" and "dog"—which would be very similar—would migrate toward one another and perhaps merge, thus allowing "cat" and "dog" to act as retrieval cues for each other, even though they may never have co-occurred. The degree of semantic relatedness of items in memory is given by the cosine of the angle between the items' context vectors (ranging from 1 for perfect synonyms to 0 for no relationship). Essentially, then, two words are closely semantically related if they appear in similar types of documents.

**Hyperspace Analogue to Language (HAL)** [edit]

Topic about HAL

The Hyperspace Analogue to Language (HAL) model[31][32] considers context only as the words that immediately surround a given word. HAL computes an NxN matrix, where N is the number of words in its lexicon, using a 10-word reading frame that moves incrementally through a corpus of text. Like in SAM (see above), any time two words are simultaneously in the frame, the association between them is increased, that is, the corresponding cell in the NxN matrix is incremented. The amount by which the association is incremented varies inversely with the distance between the two words in the frame (specifically, $\Delta = 11 - d$, where $d$ is the distance between the two words in the frame). As in LSA (see above), the semantic similarity between two words is given by the cosine of the angle between their vectors (dimension reduction may be performed on this matrix, as well). In HAL, then, two words are semantically related if they tend to appear with the same words. Note that this may hold true even when the words being compared never actually co-occur (i.e., "chicken" and "canary").

**Other Statistical Models of Semantic Memory** [edit]

The success of LSA and HAL gave birth to a whole field of statistical models of language. A more up-to-date list of such models may be found under the topic Measures of semantic relatedness.

Location of semantic memory in the brain [edit]

The cognitive neuroscience of semantic memory is a somewhat controversial issue with two dominant views.

On the one hand, many researchers and clinicians believe that semantic memory is stored by the same brain systems involved in episodic memory. These include the medial temporal lobes (MTL) and hippocampal formation. In this system, the hippocampal formation "encodes" memories, or makes it possible for memories to form at all, and the cortex stores memories after the initial encoding process is completed.

Recently, new evidence has been presented in support of a more precise interpretation of this hypothesis. The hippocampal formation includes, among other structures: the hippocampus itself, the entorhinal cortex, and the perirhinal cortex. These latter two make up the "parahippocampal cortices". Amnesics with damage to the hippocampus but some spared parahippocampal cortex were able to demonstrate some degree of intact semantic memory despite a total loss of episodic memory. This strongly suggests that encoding of information leading to semantic memory does not have its physiological basis in the hippocampus.[33]

Other researchers believe the hippocampus is only involved in episodic memory and spatial cognition. This then raises the question where semantic memory may be located. Some believe semantic memory lives in temporal neocortex. Others believe that semantic knowledge is widely distributed across all brain areas. To illustrate this latter view, consider your knowledge of dogs. Researchers holding the 'distributed semantic knowledge' view believe that your knowledge of the sound a dog makes exists in your auditory cortex, whilst your ability to recognize and imagine the visual features of a dog resides in your visual cortex. Recent evidence supports the idea that the temporal pole bilaterally is the convergence zone for

Figure 1.2: Content of the first returned page by Google

"Los Angeles Laker". We also think the relationship between "Shaquille O'Neal" and "NBA" is weaker than "Shaquille O'Neal" and "Los Angeles Laker" for some reasons like "NBA" is more general than "Los Angeles Laker". We prefer a finer relation to be stronger than coarse ones, and this is reasonable. The judgment can also be affected by other contexts. But, how can we capture this kind of features, quantify them, measure them, compare them? In fact, there are no acknowledged methods which can be utilized to quantify this kind of associations, not to mention applying them effectively into PRF. For the research topics in this thesis, the "semantic association" will actually be defined in a broad sense based on different ideas.

The key factors of PRF are two items: term association used to differentiate feedback terms and term space to restrict candidate terms we can select. Hence, to solve the problems mentioned above, we proposed several approaches based on two general ideas respectively:

**1) Discover better term semantic associations to mine feedback terms which are really related to the original query; 2) Propose methods to make a better term space so that the co-occurrence association among terms can work well. In this case, a better term space is defined as a document whose terms have a stronger semantic associations with the target information than in other documents.** Details of our proposed methods are as follows:

- The selections of feedback terms is critical to the performance of PRF, especially when the original query terms are very few. To obtain more relevant feedback terms, we take the proximity information into account. Intuitively, terms closer to the original query terms in the feedback documents are more likely to be relevant. The distance association between candidate feedback terms and the original query terms should be seriously considered in PRF. In this work, we suppose the semantic associations can be reflected from a distance among terms. We therefore proposed three methods to evaluate the strength of this kind of associations and integrate them into the classic Rocchio's model. The proposed methods are straightforward and effective on plenty of public datasets and the conclusions are helpful for other text-related research work. It is suitable for scenarios like the above "Hyperspace Analogue to Language" search example. Terms will mainly be selected from a passage around the query terms. The proposed methods are based on **general idea 1**.

- Applying proximity in PRF is straightforward and effective. However, It is hard to extend the proximity-based methods because the information behind distances is too simple to be varied.

  Recently, topic modeling becomes more and more popular in the information retrieval (IR) area. To identify how reliable a feedback document is to be relevant, we attempt to adapt the topic information into PRF. The basic idea is to determine which documents are more relevant to the subjects of the queries than others. Terms in these more relevant documents are assumed to have stronger semantic associations with the query than in other documents, or in a better term space. It is very challenging for applying the obtained topic information effectively for IR and other text-processing related areas due to the "Fuzzy Topic" obstacle. Current research work mainly focuses on mining relevant information on particular topics. However, this strategy is difficult when the boundaries of different topics are hard to be defined. In this thesis, we investigate a key factor of this problem, the topic number for topic modeling and how it makes topics "fuzzy".

  To effectively and efficiently apply topic information in PRF, we decide to take a complete view of the documents and propose a new concept "topic space". We consider a document as a mixture of different information. If we treat a document as a point in the space, we can use different coordinate systems to describe it and locate it. When we change the dimension (topic number) and the meaning of each axis of the system, we will have a different view on the document. If only some of the topics are used, we just investigate projections on some dimensions of the document, which is not complete and cannot describe the document accurately. Consequently, the performance of methods on these topics is unstable. On the contrary, when we attempt to discover useful information among documents, we always use the full topic coordinates to avoid biased topic information.

Based on this idea, we make two assumptions and propose a new framework "TopPRF" and three models TS-COS, TS-EU and TS-Entropy via integrating "Topic Space" (TS) information into PRF. These methods discover how reliable a document is to be relevant through both term and topic information. When selecting feedback terms, candidate terms in more reliable feedback documents should obtain extra weights. The measurement of "relevance" is based on how a feedback document is semantically similar to the sample documents. The latter are assumed to be completely reliable, and we treat these sample documents as the representation of the target information for queries. Experimental results on various public collections justify that our proposed methods can significantly r educe the influence of “fuzzy topics” and obtain stably good results over the strong baseline models. Our proposed framework TopPRF and the three topic-space based methods can search documents beyond traditional term matching only, and provide a promising avenue for constructing better topic space-based IR systems. Moreover, in-depth discussions and conclusions are made to help other researchers apply topic information effectively. The proposed methods are based on **general idea 2**.

- We also proposed an adaptive co-training method to obtain feedback documents of high quality. In a similar way as the previous one, we assume each one of a small group of documents can improve the overall performance of PRF (in good quality), and they can represent the target information users need. Also, we have another small group of “bad quality” documents. Instead of measuring the similarity association directly, in this work, we apply methods to learn which documents are semantically similar to the examples and then identify more good ones. Only documents in good quality will be used for PRF. The biggest obstacle of these applications is the lack of training data. Because human efforts are not applied in PRF methods, the top k feedback documents are considered to be relevant and used as training data. Mostly, $k$ is a number smaller than 50 to ensure the quality of learning data, but that is not enough for traditional learning methods. To deal with this situation, we utilized the co-training method to mine good feedback documents which can improve the overall performance of PRF. Co-training is a semi-supervised learning technique that assumes that each example is described using two different feature sets. The two sets provide different, complementary information about the instance. Co-training first learns a separate classifier for each view using any labeled examples. The most confident predictions of each classifier on the unlabeled data are then used to iteratively construct additional labeled training data (In our proposed method, the semi-supervised co-training is revised to be an unsupervised one because no real labeled data are needed at all). Co-training only requires small amounts of labeled data that top $k$ and bottom $v$ documents obtained from the first-pass retrieval are labeled to be "good" and "bad" samples for two classifiers. For the co-training process, the number of iterations is determined by users so it cannot be optimal for different data. Sometimes, the learned classifier does not work well. To this end, we take the quality of the learned classifiers into account and make the co-training method more adaptive. Extensive experiments justify the effectiveness of our adaptive co-training method. The adaptive co-training method is based on **general idea 2**.

- The last work is not directly related to the research purpose of applying semantic associations to improve PRF. However, it is vital and has long-term benefits for the research community. In the research of IR, researchers often need to make a comparative study to evaluate their ideas. Sometimes, however, it is not easy to obtain a reliable baseline while the study is new. Also, to make progress in this area, it is better to have a framework to adapt new techniques. Since there are many techniques, how to categorize and integrate them in the framework is challenging. Thus we propose a new hybrid model which can simply and flexibly combine components of three different IR techniques under a uniform framework, including the proximity factor. Other factors can also be adapted, which need much more experiments and literature reviews. The work is promising but complex, so we expect more attempts to be done in the future.

## 1.4 Main Contributions of the Thesis

This thesis contributes to the relevance feedback area of information retrieval. Specifically, it introduces novel ideas and techniques to the fields of pseudo relevance feedback. It investigates how to adapt semantic associations into PRF to obtain good and robust performance. The main contributions of the research work in this thesis are as follows:

1. We measure the strength of semantic associations among candidate feedback terms and queries through proximity. The proximity information is firstly utilized in the probabilistic framework for PRF. Three new methods are proposed to convert the proximity of terms into term associations so that relevant feedback terms can be discriminated from irrelevant terms for PRF.

2. To be more important, we extend the work of mining semantic associations among terms from the simple proximity to topics. To identify which feedback documents is more reliable to be semantically relevant, we take the topic information from feedback documents into account. Before the proposal of our methods, we first investigate the "fuzzy topic" obstacle and provide evidence to justify how it makes the performance of topic applications unstable, especially for methods rely on particular topics. Then we provide a novel way to make full use of topic information to obtain more robust and better performance and propose a new framework TopPRF by introducing a new concept "topic space". To identify which documents are more reliable than others, we need to rank the documents according to their scores. By our methods, the relative ranks of feedback documents according to their scores (e.g., co-

sine similarity scores) are very stable. No matter how the topic distributions change, terms in those highly ranked documents will be consistently more important than others for query expansion. This is an important discovery for applying topic information for IR, especially when there is not an optimal topic number for corpora. This is a further step beyond the term applications because the usage of topics in PRF is more relative to the concept "semantic".

3. We also propose an adaptive co-training method to solve the lack-of-training-data problem for applying machine learning methods for PRF. Besides measuring the semantic associations among terms/documents directly, using methods to learn which documents are more similar to the sample ones is a different point of view, and it has been proven to be effective as well. Even several pseudo feedback documents can help to improve the overall performance of PRF. While machine learning techniques are used in IR more and more, the adaptive idea is useful for other machine learning methods without adequate training data.

4. We attempt to build a hybrid framework and adapt new technologies so that researchers can continuously make progress in the IR area. Mainly, we choose three kinds of techniques and use them in our proposed framework. We add these methods as the components one by one and demonstrate how much improvement each of them can make. Also, we compare the best performance obtained through our

framework with the best TREC performance in 9 years and find that our proposed work is usually better. This work, therefore, can provide a very strong baseline for further studies, especially when it 's hard to make fair comparisons as a new study is conducted. A new technique like different semantic associations which can make improvements over our framework can solidly benefit the community and attract more attention.

Generally, we propose several effective and novel methods to improve the performance of PRF significantly based on the two general ideas. When investigating how to measure the semantic associations in corpora and utilize them to enhance PRF, we discover factors like proximity, document quality which is beneficial. Beyond term information, topic information can also help and make the association we find more "semantic". The work proposed in this thesis can inspire people to discover more factors which can build a better query, integrate them into a uniformed framework and make solid progress in the IR area[4].

[4]According to the comments from my committee Prof. Jamie Callen, "PRF does very well when measured by mean average precision (MAP), because the relative change metric favours improved queries over degraded queries. However, PRF does not do well when measured by win/loss ratios. Commercial systems are leery about using a technique that hurts 30% of the queries." To be consistent with the previous research, we still use MAP as the main metric for comparisons and evaluations.

## 1.5 Outline

The rest of this thesis is organized as follows. Chapter 2 describes related work including background knowledge about basic models, PRF models, co-training, proximity, topic modeling and their applications respectively. Chapter 3 introduces how proximity information among terms is used to evaluate the relevance of candidate feedback terms. Chapter 4 introduces a new concept “topic space” and how it is utilized to measure the reliability of feedback documents and then use the results to adjust the weights of candidate feedback terms. An adaptive co-training PRF method and a hybrid framework for integrating different state-of-the-arts IR components are presented in Chapter 5 and Chapter 6 respectively. Chapter 7 summarizes features of the work in this thesis and makes a comparison of the proposed methods. Chapter 8 concludes this thesis and discusses potential future work.

# Chapter 2

# Related Work

## 2.1 Basic Models

Basic models are used to weight the documents in the collections and obtain the initial document list for PRF. The process of retrieving the initial document list is called *first-pass retrieval.* Generally, traditional models for IR can be classified into two categories: probabilistic models and language models. The former estimate the probability of relevance for a document directly based on some particular probability distributions; the latter treat both documents and queries as language models and estimate how likely a document language model can generate the query or vice versa. Our work is all based on probabilistic models, and we use state-of-the-art language models for comparisons.

### 2.1.1 BM25

BM25 [RWJ$^+$94b] is one of the best weighting models for IR. It is a probabilistic model, and we use it as the basic model for our work. In BM25, a search term is assigned a weight based on its within-document term frequency and query term frequency, and the document weight is the sum of the query term weights in it. The corresponding weighting function is as follows:

$$w = \frac{(k_1 + 1) * tf}{K + tf} * \log \frac{(N - n + 0.5)}{(n - 0.5)} * \frac{(k_3 + 1) * qtf}{k_3 + qtf} \tag{2.1}$$

where $w$ is the weight of a query term, $N$ is the number of indexed documents in the data set, $n$ is the number of documents containing a specific term, $tf$ is within-document term frequency, $qtf$ is within-query term frequency, $dl$ is the length of the document, $avdl$ is the average document length, $nq$ is the number of query terms, the $k_i$s are tuning constants, $K$ equals to $k_1 * ((1 - b) + b * dl/avdl)$.

### 2.1.2 Language Models

Given a query $q = q_1 q_2 ... q_n$ and $q_i$ is a query term, language modeling assumes the weight of a document is conditional probability $p(d|q)$ and $p(d)$ has a uniform distribution. So $p(d|q)$ can be calculated as follow:

$$p(d|q) \propto p(q|d)p(d) \tag{2.2}$$

Unigram language models are very popular basic models. They assume terms are independent to each other give a document. So $p(q|d) = \prod_i p(q_i|d)$. The unigram language model estimates the weight of a term by combining its document frequency and collection frequency. The unigram language model with Dirichlet smoothing is proven to perform well [ZL04]. It calculates the weight of term by the following equation:

$$p(w|d) = \frac{c(w;d) + \mu p(w|C)}{\sum_w c(w;d) + \mu} \tag{2.3}$$

where $w$ is a term, $c(w;d)$ is the word count of the term $w$ in the document $d$, $p(w|C)$ is the collection language model, $\mu$ is the smoothing parameter

## 2.2 Pseudo Relevance Feedback

In information retrieval, PRF via query expansion is referred to as the techniques, algorithms or methodologies that reformulate the original query by adding new terms into the query, to achieve a better retrieval performance. There are a large number of studies on the topic of PRF. Here we mainly review the work about PRF which is the most related to our research.

### 2.2.1 Rocchio's Model

Rocchio's algorithm [Roc71a] is a classic framework for implementing (pseudo) relevance feedback via improving the query representation. When the negative feedback documents are ignored, the traditional Rocchio's model is as follows:

$$\vec{Q_1} = \alpha * \vec{Q_0} + \beta * \sum_{r \in R} \frac{\vec{r}}{|R|} \tag{2.4}$$

where $Q_0$ and $Q_1$ represent the original and first iteration query vectors which contain the weights of all the terms, top terms in $Q_1$ will be chosen as feedback terms, $R$ is the set of (pseudo) relevance documents, and $r$ is the expansion term weight vector.

Although the Rocchio's model has been introduced for many years, it is still effective in obtaining relevant documents. According to [Zha08b], "BM25 term weighting coupled with Rocchio feedback remains a strong baseline which is at least as competitive as any language modeling approach for many tasks". Meanwhile, it is very suitable for researchers to extend it with extra information factors.

### 2.2.2 Relevance Models

For PRF in the language modeling framework, we always exploit feedback information (e.g., the top-ranked documents set, $F = D_1, D_2, \ldots, D_{|F|}$), in order to re-estimate a more accurate query language model. For example, the model-based feedback approach [ZL01b] is not only theoretically sound, but also performs well empirically. The essence of model-based feedback is to update the probability of a term in the query language model by making use of the feedback information. Much like model-based feedback, relevance models [LC01a] also estimate an improved query model. The difference between the two approaches is that relevance models do not explicitly model the relevant or pseudo-relevant document. Instead, they model a more generalized notion of relevance [MC07]. Lv *et al.* [LZ09b] have conducted a comparable study of five representative state-of-the-art methods for estimating improved query language models in ad hoc information retrieval, including RM3 (a variant of the relevance language model), RM4, DMM, SMM (a variant of model-based feedback approach), and RMM [LC01a, TZ06, ZL01b]. They found that SMM and RM3 are the most effective in their experiments, and RM3 is more robust to the setting of feedback parameters. So we use RM3 as a baseline for our research in this thesis.

Relevance language models do not explicitly model the relevant or pseudo-relevant document. Instead, they model a more generalized notion of relevance $R$. The formula of RM1 is:

$$p(w|R) \propto \sum_{\theta_D} p(w|\theta_D)p(\theta_D)P(Q|\theta_D) \tag{2.5}$$

The relevance model $p(w|R)$ is often used to estimate the feedback language model $\theta_F$, and then interpolated with the original query model $\theta_Q$ in order to improve its estimation as follows:

$$\theta_{Q'} = (1-\alpha) * \theta_Q + \alpha * \theta_F \tag{2.6}$$

This interpolated version of relevance model is called `RM3`.

## 2.3 Co-training methods

Co-training using unlabeled data has shown it effectiveness for reducing error rates in text classification [BM98, GZ00, Joa99, NMTM00, ZH01]. The idea of co-training is originally proposed by Blum & Mitchell for boosting the learning performance when there is only a small amount of labeled examples available [BM98]. Under the assumption that there are two redundant but not completely correlated views of an example, unlabeled data are shown to be able to augment labeled data [BM98]. Co-training is also used for extracting knowledge from the World Wide Web [CDF$^+$00], or for email classification [KMAH04]. The result shows that it can reduce the classification error by a factor of two using only unlabeled data. However, the performance of co-training depends on the learning methods used [CDF$^+$00, KMAH04].

There are some other studies that explore the potentials of co-training in recent years. Modified versions of the co-training method have been proposed. [GZ00] use two different supervised learning algorithms to label data for each other. Raskutti presents a new co-training strategy which uses two feature sets. One classifier is trained using data from original feature space, while the other is trained with new features that are derived by clustering both the labeled and unlabeled data [RFK02].

All the above work shows that the co-training method can be used to boost the performance of a learning algorithm when there is only a relatively small set of labeled examples given. However, this idea is based on the assumption that the dataset comes with two sets of features which are distinct in nature, but this is not the case in document retrieval. [CKP04] suggest to randomly split one single feature set into two sets for co-training. Their experimental results show that a random split of the feature set leads to comparable, and sometimes, even better results than using the natural feature sets. [HHW$^+$06a] investigate the effect of Chan et al.'s version of co-training on the TREC HARD track data for passage retrieval by employing a dual-index model for term weighting. The main work of Huang et al. (2006) applies co-training to identify more relevant passages based on a small set of training data and use the results to re-estimate parameters of the probabilistic weighting function, BM25.

Recently, there have been efforts in applying machine learning methods to find useful feedback documents or expansion terms. Lee et al. apply a clustering-based resampling method to select good feedback documents based on the relevance model. Relevance density is utilized to evaluate the quality of feedback documents. Cao et al. use features such as the proximity of expansion terms to the query terms, the expansion terms and query terms co-occurrences, etc. to predict which expansion terms are useful [CNGR08]. Based on similar features, He & Ounis select good feedback documents using standard machine learning algorithms [HO09a]. All methods proposed in [LCA08], [CNGR08] and [HO09a] provide a moderate success over a traditional QE baseline.

## 2.4 Proximity used in IR

Term proximity is the co-occurrences of terms within a specified distance. Particularly, the distance is the number of intermediate terms in a document. Plenty of work has been done to integrate term proximity into both probabilistic and language models. Keen [Kee91, Kee92] attempted to import term proximity in the Boolean retrieval model by introducing a "NEAR" operator. Buttcher *et al.* [BCL06] proposed an integration of term proximity scoring into Okapi BM25 and obtain improvements on several collections. Rasolofo *et al.* [RS03] added additional weight to the top documents which contain query term pairs appearing in a window through a two-phase process, but the improvement is somewhat marginal. Song *et al.* [STW$^+$08] presented a new perspective on term proximity. Query terms are grouped into phrases, and the contribution of a term is determined by how many query terms appear in the context phrases. To make it clear that how we could model proximity and incorporate a proximity measure, Tao *et al.* [TZ07] systemically studied five proximity measures and investigated how they perform in the KL-divergence retrieval model and the Okapi BM25 retrieval model. Under the language modeling framework, Zhao *et al.* [ZY09] used a query term's proximate centrality as a hyper parameter in the Dirichlet language model. Lv *et al.* [LZ09d] integrated the positional and proximity information into the language model by a different way. They defined a positional language model at each position in documents by create virtual documents based on term probation.

All the above work focuses on how to utilize the proximity information of query terms to avoid documents which contain scattered query terms. This kind of documents should be punished because they are very likely to be irrelevant. For example, a document contains both "Japan" and "Earthquake" is possible to be unrelated to the topic "Earthquake in Japan" if these two terms are not close in the context. It could be biased to only "Earthquake" and mention some technologies in "Japan" about "Earthquake". Term proximity is effective to discriminate against these types of documents. Although there have been plenty of efforts in integrating proximity heuristic into existing retrieval models, research work about how to utilize this information for pseudo relevance feedback is still limited. Vechtomova *et al.* [VW06] combined several distance factors with Mutual Information for selecting query expansion terms from windows or passages surrounding query term occurrences. However, marginal improvements were observed in the experiments. Lv *et al.* [LZ10] presented two methods to estimate the joint probability of a term $w$ with the query $Q$ at every position in each feedback document. This is an extension of the state-of-the-art relevance model [LC01a], and significant improvements were obtained on two collections. Besides the work presented [LZ10, VW06], it is difficult to find other systematical work about formally modeling term proximity heuristic in the context of pseudo feedback, especially in the classic Rocchio's model.

## 2.5 Topic Modeling

Recently, probabilistic topic models are becoming more and more popular in the text mining area [Hof99, BNJ03a, LM06]. This kind of models can discover the possible semantic schemes in a group of documents. The basic idea of topics models is that the vocabulary of a document is generated from topics, and topics are represented as different probability distributions of terms in the vocabulary. A term can have various probabilities on different topics.

Probabilistic Latent Semantic Indexing (PLSI) [Hof99] is a significant step in the development of topic models. It models each word in a document as a sample from a mixture model, where the mixture components are multinomial random variables that can be viewed as representations of "topics". Thus each word is generated from a single topic, and different words in a document may be generated from different topics. Each document is represented as a list of mixing proportions for these mixture components and thereby reduced to a probability distribution on a set of topics. Latent Dirichlet Allocation (LDA) [BNJ03a] is another popular topic model which also assumes that there are topics in the corpus, but a document can have more than one topic. LDA has a more complicated probabilistic procedure of generating a document. Another state-of-the-art topic model named Pachinko Allocation Model (PAM) [LM06] was proposed in 2006. Unlike PLSI and LDA, topics are not considered to be independent. The four-level PAM models utilize a super-topic layer in an adirected acyclic graph to model the correlations among topics.

## 2.6 Topic Modeling for PRF

Topic models have been applied in PRF recently, but not all of the applications are effective. Yi and Allan [YA08] attempted to use different topic models for the retrieval purpose, and the proposed PRF methods CBQE, LBQE and PBQE were all worse than the state-of-the-art RM method. LDA-RM in [YA09] can outperform RM in some cases, but cannot obtain sustainable improvements over the latter. Andrzejewski et al. [AB11] utilized LDA to generate latent topics and decided which latent topics are potentially relevant. Users select a latent topic from them. Then vocabulary terms which are most strongly associated with that topic are used to augment the original query. Ye and Huang [YHL11b] have proposed three methods to obtain the most relevant topics and select feedback terms from them. Significant improvements have been made and they found their proposed methods performed much better in the simulated relevance feedback. Their research work indicates that the performance of topic model based PRF methods depends on the quality of corpus where topics are obtained. However, when the topic number changes, the performance of the proposed Top_k method drops significantly. [CA12] incorporated topic information of relevant documents and irrelevant documents in active relevance feedback and obtain promising performance on the medical OHSUMED dataset. Their research shows that topic information can be useful in domain-specific search. [WZWS12] assumed terms should be in relevant topics before and after being translated and used LDA-based PRF for the cross-language retrieval task. They used the same strategy to select topics as in [YHL11b] and obtained marginal improvements. [SK13] found that the precision of topic-based relevance feedback method can be better than the word-based relevance feedback model in particular cases, but the overall performance is not satisfying.

To the best of our knowledge, previous work on applying topic models in IR focused mainly on how to find the most relevant topic(s) and did not concern the fuzzy topic problem [AB11, YHL11b, WZWS12, SK13]. In fact, the problem is very important and has an extensive impact on the relevant topics they pursue. Some work is on human-involved relevance feedback [AB11, CA12]. This makes the utilization of topic models very expensive and time-consuming. Also, only little study is about integrating topic information into the probabilistic PRF model, and almost all the studies import extra parameters for interpolations. While topic modeling is complicated, more parameters will increase the complexity and therefore cost more computing resources.

Generally, the research on PRF has been carried on for a long time. The identification of query terms which can help improve the overall performance is still a challenge. As far as we know, there is not a widely-accepted criterion to precisely, separate terms will be beneficial to original queries from others. Most, if not all previous research is still evaluated based on the statistical results of hundreds of queries. Even a new method can make improvements over the state-of-the-art baselines; it can also bring noise information into queries while documents are various in qualities, lengths. Some queries are better, and some queries are worse. To some extent, the way of making a better query is not of fine-granularity. Researchers are still in the stage of discovering what kind of features can help PRF. So it remains a challenging problem for my further research.

# Chapter 3

# Proximity-based Rocchio's Model for Pseudo Relevance Feedback

Term proximity is an effective measure for the strength of term associations, which has been studied extensively in the past few years. Most of these studies focused on the term proximity within the original query and adapt this in ranking documents [BCL06, CCT00, HHZ11a, Kee91, LZ09d, RS03, STW$^+$08, TZ07]. Various methods of integrating proximity information into a retrieval process are introduced in the previous work, and it has proven to be useful in discriminating between the relevant and non-relevant documents.

In the field of PRF, based on the assumption that terms closer to the query terms are more likely to be relevant to the query topic, there are several studies which investigated how to give more weight to these terms in the process of pseudo relevance feedback. For example, to this end, Vechtomova *et al.* [VW06] imported a distance factor which combined with Mutual Information (MI) for selecting query expansion terms. Lv *et al.* [LZ10] proposed a positional relevance model (PRM) by using position and proximity information to solve this problem and obtain significant performance.

However, as far as we are aware, there is little work done on incorporating proximity information into the traditional Rocchio's feedback model. Although the Rocchio's model has been introduced in the information retrieval field for many years, it is still effective in obtaining relevant documents. This observation is also supported in our preliminary experiments of this work. Therefore, it is promising to make an extension of the Rocchio's model to take into account the proximity information.

In addition, it is unknown how to tackle the challenge of modeling the traditional statistics of expansion terms and the relationship between expansion terms and the query terms in a unified framework. In this thesis, we propose a proximity-based feedback model based on the traditional Rocchio's model, called `PRoc`. Unlike another language model with the similar idea, positional relevance model (PRM), we focus on the proximity of terms rather than the positional information. In our method, we estimate the weights of candidate expansion terms by taking their distance from query terms into account. Specifically, if a term is far away from the query terms in the feedback documents, it is proposed to be punished by discounting its weight because it is likely to be irrelevant to the query topic.

Although there are various methods to model proximity, it is still uncertain which one is the best for a particular retrieval model. Since no previous work has been done on the Rocchio's model, we try three methods to convert the proximity into term associations, measure the strength of these associations and estimate the importance of candidate feedback terms. We hope this work can provide meaningful baselines for other researchers when integrating term proximity into the state-of-the-art probabilistic pseudo relevance feedback models.

## 3.1 Proximity-based Rocchio's Model

In this section, we present the proposed proximity-based Rocchio's model, called PRoc. Specifically, we first briefly introduce more details about the traditional Rocchio's model than in the Related Work chapter, and present the adoption of Rocchio's model for proximity information by proposing a new concept, namely proximity-based term frequency ($ptf$). Then we describe in details about how to adopt $ptf$ in three investigated proximity measures.

### 3.1.1 Adaption of Rocchio's Model

Rocchio's model [Roc71b] is a classic framework for implementing (pseudo) relevance feedback via improving the query representation. It models a way of incorporating (pseudo) relevance feedback information into the vector space model (VSM) in IR. In case of pseudo relevance feedback, the Rocchio's model without considering negative feedback documents has the following steps:

1. All documents are ranked for the given query using a particular retrieval model. This step is called *first-pass retrieval*, from which the $|R|$ highest ranked documents are used as the feedback set.

2. Each document in the feedback set $R$ is represented as a weighted term vector, annotated by $r$, by a certain weighting function, for example originally by the TFIDF weights [Sal71].

3. The representation of the query is finally refined by taking a linear combination of the initial query term vector with the feedback document vector, which is introduced in Chapter 2 in details:

$$Q_1 = \alpha * Q_0 + \beta * \sum_{r \in R} \frac{r}{|R|} \tag{3.1}$$

Many other relevance feedback techniques and algorithms [Ama03b, CCB01, RWHB$^{+}$95] are also derived under the Rocchio's framework. For example, Carpineto *et al.* proposed to compute the weight of candidate expansion terms based on the divergence between the probability distributions of terms in the top ranked documents and the whole collection. In this work, we also take advantage of this distributional view. But we re-interpret the definition of *term frequency* in the KLD formula 3.2 instead of the distribution estimated from a set of top documents. We use the following function to rank the candidate terms:

$$score(w) = P(w|d) * log(\frac{P(w|d)}{P(w|C)}) \tag{3.2}$$

where $P(w|d)$ is the probability of candidate expansion term $w$ in feedback document $d$, $P(w|C)$ is the probability in the retrieval collection $C$.

Traditionally, candidate terms are ranked by their weights in the feedback documents, and the weights are affected by term frequencies in these documents extensively. However, the normal term frequency cannot capture the characteristic that whether a candidate term occurs near or far away from the query, such that the candidate term may not be relevant to the query topic. In other words, if the occurrence of a term is far away from the query terms, it should not be counted in the effective term frequency because this term is very likely to be irrelevant to the query topic. Thus, we propose a new concept, proximity term frequency (*ptf*), which models the frequency of a term as well as the semantics of the query regarding proximity. To adapt the proximity information, we re-interpret the definition of *term frequency* by proposing three kinds of proximity measures: window-based method, kernel-based method and the hyperspace analog to language method. The main research challenge now we are facing is how to evaluate *ptf*. In the following subsections, we introduce three measures to compute the $ptf$. Meanwhile, the importance of query terms is also taken into account. A very frequent query term is likely to be close to many candidate terms, which makes it difficult to distinguish the related feedback terms. Inverse document frequency (*idf*) of query terms is integrated to calculate $ptf$.

### 3.1.2 Window-based Method

The first method adopts a simple window-based n-gram frequency counting method, which has been popular in previous studies on using term proximity for IR (e.g. [MC05, PO07]).

The basic idea of the window-based n-gram counting method is to segment the document into a list of sliding windows, with each window having a fixed window size *wSize*. If a document has a length of *l*, and the window size is set to *wSize*, the document is then segmented into *l-wSize* sliding windows, where each window contains *wSize* consecutive tokens. For example, if a document has four tokens A, B, C, and D, and the window size is 3, there are two windows in this document, namely A, B, C and B, C, D. The n-gram frequency is then defined as the number of windows in which all n-gram terms co-occur. There could be two variants of the n-gram models, namely the *ordered* and *unordered* n-gram models. The ordered n-gram model takes the order of occurrences of the n-gram terms into account. For the same n-gram terms, the n-grams in which the composing n-gram terms appear in different orders are considered as different n-grams. In contrast, the unordered model ignores the order of occurrences of the n-gram terms. Actually, only a rough distance between terms is considered in this measure. If two terms are in the window, they are strongly related and the co-occurrence is counted in $ptf$.

$$ptf(t) = \sum_{i=1}^{|Q|} C(t, q_i) IDF(q_i) \tag{3.3}$$

where $q_i$ is a query term, C(t, $q_i$) is the number of windows in which the candidate term and the query terms co-occur, $|Q|$ is the number of query terms, and $IDF(q_i)$ equals to $log(N - N_t + 0.5)/(N_t + 0.5)$. $N$ is the number of documents in the collection, and $N_t$ is the number of documents that contain $q_i$.

The n-gram counting method has the advantage of being straightforward, and can be easily deployed in practice. It does not take the actual distance between query terms into account directly, and any n-gram terms appear together within a window is counted as one occurrence of the n-gram. If a term is very close to a query term, its co-occurrence count with the query term will be more than that of a term far away from this query term in the sliding windows. This variant of PRoc is denoted by `PRoc1` in the rest of this work.

### 3.1.3 Kernel-based Method

Following previous studies [LZ10, ZHH11], an alternative method we use is a kernel-based method to count the term frequency in a document. There are some kernel functions (e.g. Gaussian, Triangle, Cosine, and Circle [ZHH11]) which were used for measuring the proximity. Gaussian kernel has been shown to be effective in most cases. In this work, we also use the Gaussian kernel to measure the proximity between a candidate expansion term $t$ and a query term $q$.

$$K(t, q) = exp[\frac{-(p_t - p_q)^2}{2\sigma^2}] \tag{3.4}$$

where $p_t$ and $p_q$ are respectively the positions of candidate term $t$ and query term $q$ in a document [5], $\sigma$ is a tuning parameter which controls the scale of Gaussian distribution. In other words, $\sigma$ has a similar effect as the parameter $wSize$ in the window-based method. To keep the consistency with other proximity measures, we also use $wSize$ to denote $\sigma$. Different from the window-based method, the kernel-based method is a soft proximity measure. In particular, even if the appearance of a candidate term and a query term is not in a window of $wSize$, its weight can still be slightly boosted.

In this method, besides the average proximity to the query, we also take into

[5] $p_t$ and $p_q$ are positions, not position vectors

account the importance of different query terms. Therefore, we build a representational vector for the query, in which each dimension is the weight of a query term by the inverse document frequency formula below, and then the proximity-based term frequency $ptf$ in the Kernel-based method is computed as follows:

$$ptf(t) = \sum_{i=1}^{|Q|} K(t, q_i) IDF(q_i) \tag{3.5}$$

where $q_i$ is a query term, $|Q|$ is the number of query terms, and $IDF(q_i)$ is the same as in PRoc1. $N$ is the number of documents in the collection, and $N_t$ is the number of documents that contain $q_i$. The second variant of PRoc is denoted by `PRoc2` in the rest of this work.

### 3.1.4 HAL Method

The Hyperspace Analogue to Language (HAL) [KL95] is a computational modeling of psychological theory of word meaning by considering context only as the words that immediately surround the given the word. The basic motivation is that when a human encounters a new concept, its meaning is derived via other concepts that occurr within the same context.

As shown in [YHL11c], the HAL Space is automatically built from a corpus of text, defined as follows: for each term in a specified vocabulary $V$, a $|V| \times |V|$ matrix is built by moving a sliding window of length $wSize$ across the corpus, where $|V|$ is the number of terms in vocabulary $V$. All words within the window are considered as co-occurring with each other with strengths inversely proportional to the distance between them. The weightings of each pair of co-occurring terms are accumulated over the corpus. Then, a term can be represented by a semantic vector, in which each dimension is the weight for this term and other terms as follows:

$$HAL(t'||t) = \sum_{k=1}^{wSize} w(k)n(t,k,t') \tag{3.6}$$

where $k$ is the distance from term $t'$ to $t$, n(t, k, t') is the co-occurrence frequency within the sliding windows when the distance equals $k$, and $w(k) = wSize - k + 1$ denotes the strength.

In this work, we adapt the original HAL model similarly as in [KL95]. In particular, to measure the proximity between a candidate expansion term and the original query, we restrict the context to the query terms, not all the co-occurring terms in the feedback documents. With this adoption, the resulting vector for each candidate term denotes a proximity relationship with the entire query. Like the Kernel-based method, we also take into account the importance factor of query terms in the same way. Then, the HAL based $ptf$ is as follows:

$$ptf(t) = vec(t) \cdot vec(Q) = \sum_{i=1}^{|Q|} HAL(t||q_i)IDF(q_i) \tag{3.7}$$

$IDF(q_i)$ is the as in PRoc1

In the adoption of proximity information in PRF, $ptf$ replaces the traditional term frequency in our approach.

The weighted HAL model includes the information of term distances and co-occurrence frequencies completely. It is the first time that this linguistic model is adopted to measure the proximity. The third variant of PRoc is denoted by `PRoc3` in the rest of this work.

Generally, $ptf$ adjusts the traditional term frequency by integrating proximity. This help identify terms which are more related to the original queries through a very simple and effective way. The concept $ptf$ can be easily adopted into other text mining or retrieval applications.

## 3.2 Experimental Settings

### 3.2.1 Test Collections

In this section, we describe four representative test collections used in our experiments: Disk4&5, WT2G, WT10G, and GOV2, which are different in size and genre. The Disk4&5 collection contains newswire articles from various sources, such as Association Press (AP), Wall Street Journal (WSJ), Financial Times (FT), etc., which are usually considered as high-quality text data with little noise. The WT2G collection is a general Web crawl of Web documents, which has 2 Gigabytes of uncompressed data. This collection was used in the TREC 8 Web track. The WT10G collection is a medium size crawl of Web documents, which was used in the TREC 9 and 10 Web tracks [6]. It contains 10 Gigabytes of uncompressed data.

The GOV2 collection, which has 426 Gigabytes of uncompressed data, is crawled from the .gov domain. This collection has been employed in the TREC 2004, 2005 and 2006 Terabyte tracks. GOV2 is a very large crawl of the *.gov* domain, which has more than 25 million documents with an uncompressed size of 423 Gigabytes. There are 150 ad-hoc query topics, from TREC 2004 - 2006 Terabyte tracks, associated to GOV2. In our experiments, we use 100 topics in TREC 2005 - 2006.

The TREC tasks and topic numbers associated with each collection are presented in Table 3.1. As we can see from this table, we evaluate the proposed approach with a relatively large number of queries.

In all the experiments, we only use the *title field* of the TREC queries for retrieval. It is closer to the actual queries used in the real application and feedback is expected to be the most useful for short queries [ZL01b].

In the process of indexing and querying, each term is stemmed using Porter's English stemmer [Por80], and stopwords from InQuery's standard stoplist [ACC$^+$00] with 418 stopwords are removed. MAP (Mean Average Precision) of the top 1000

[6]WT2G is very closely related to WT10g, so those two datasets probably don't give independent results

Table 3.1: The TREC tasks and topic numbers associated with each collection.

| Collection | Task | Queries | Docs |
|---|---|---|---|
| Disk4&5 | TREC 2004, Robust | 301-450 | 528,155 |
| WT2G | TREC8, Web ad-hoc | 401-450 | 247,491 |
| WT10G | TREC9, 10, Web ad-hoc | 451-550 | 1,692,096 |
| GOV2 | TREC04-06, Web ad-hoc | 701-850 | 25,178,548 |

documents is used as the evaluation metric, as is commonly done in TREC evaluations. The MAP metric reflects the overall accuracy and the detailed descriptions for MAP can be found in [VH00]. We take this metric as the primary single summary performance for the experiments, which is also the main official metric in the corresponding TREC evaluations.

### 3.2.2 Baseline Models

In our experiments, we compare our PRoc models with the traditional combination of BM25 and Rocchio's feedback model. In addition, we compare the proposed models with the state-of-the-art feedback models in the KL-divergence language modeling (LM) retrieval framework. In particular, for the basic language model, we use a Dirichlet prior (with a hyperparameter of $\mu$) for smoothing the document language model as shown in Equation 2.3.

We train the parameters in the document language model in all the experiments in order to make fair comparisons, and focus on evaluating different ways of approaching the query-related topic for PRF.

For PRF in language modeling framework, we first compare our proposed model with the relevance language model [LC01a, LZ09b], which is a representative and State-Of-The-Art approach for re-estimating query language models for PRF [LZ09b]. Lv *et al.* [LZ09b] systematically compared five state-of-the-art approaches for estimating query language models in ad-hoc retrieval, in which RM3 not only yields impressive retrieval performance in both precision and recall metric, but also performs steadily. The equation of RM3 can be found in Section 2.2.2. In particular, we apply Dirichlet prior for smoothing document language models [ZL01b].

### 3.2.3 Parameter Settings

As we can see from all the PRF retrieval models in our experiments, there are several controlling parameters to tune. In order to find the optimal parameter setting for fair comparisons, we use the training method presented in [DM06] for both the baselines and our proposed models, which is popular in the IR domain for building strong baselines. In particular, first, for the smoothing parameter $\mu$ in LM with Dirichlet prior, we sweep over values from 500 to 2000 with an interval of 100. Meanwhile, we sweep the values of $b$ for BM25 from 0 to 1.0 with an interval of 0.1. Second, for parameters in PRF models, we empirically set the number of top documents to 20 in baseline PRF approaches and our PRoc models, the number of expansion terms ($k \in 10, 20, 30, 50$), and the interpolation parameter ($\beta \in 0.0, 0.1, \ldots, 1.0$). The window size for PRoc models is from 10 to 1500 with an interval 10. To evaluate the baselines and our proposed approach, we use 2-fold cross-validation, in which the TREC queries are partitioned into two sets by the parity of their numbers on each collection. Then, the parameters learned on the training set are applied to the test set for evaluation purpose as in [MNCR09].

## 3.3 Experiments and analysis

### 3.3.1 Comparison of Basic Retrieval Models. Topics for these experiments are as shown in Table 3.1

As we mentioned in the previous section, the results of both models are obtained by 2-fold cross-validation. Therefore, it is fair to compare them on these four collections. BM25 slightly outperforms LM with Dirichlet prior on the WT2G collection. The results of these two models are almost the same over the Disk4&5, WT10G and GOV2 collections. This comparison indicates that the classic BM25 model is generally

Table 3.2: BM25 vs LM on the four TREC collections

| | disk4&5 | WT2G | WT10G | GOV2 |
|---|---|---|---|---|
| BM25 | 0.2216 | 0.3124 | 0.2055 | 0.3034 |
| LM | 0.2247 | 0.2995 | 0.2063 | 0.3040 |

Table 3.3: PRoc compares with BM25+Rocchio and LM+RM3 on Disk4&5. The percentages in the parenthesis are the improvement gains over the classic Rocchio's model and RM3. A "*" indicates a statistically significant improvement over the classic Rocchio's model baseline, and a "+" indicates a statistically significant improvement over the RM3 model baseline according to the Wilcoxon matched-pairs signed-ranks test at the 0.05 level. The bold phase style means that it is the best result.

| # of feeback terms | PRoc1 | PRoc2 | PRoc3 | BM25 + Rocchio | LM + RM3 |
|---|---|---|---|---|---|
| 10 | $0.2509^{+}$ | $0.2523^{+}$ | **0.2571**$^{*+}$ | 0.2463 | 0.2289 |
| 20 | $0.2567^{+}$ | $0.2596^{+}$ | **0.2647**$^{*+}$ | 0.2504 | 0.2326 |
| 30 | $0.2589^{+}$ | $0.2595^{+}$ | **0.2662**$^{*+}$ | 0.2545 | 0.2356 |
| 50 | $0.2578^{+}$ | $0.2602^{+}$ | **0.2662**$^{*+}$ | 0.2533 | 0.2382 |
| Average | 0.2561 (2.00%, 9.54%,) | 0.2579 (2.71%, 10.31%,) | 0.2636 (4.98%, 12.75%,) | 0.2511 | 0.2338 |

comparative to LM, and it is reasonable to use them as the basic models of the PRF baselines and our proposed model.

### 3.3.2 Comparison with PRF Models

All the experimental results can be found in the four tables. From Table 4.3 to Table 3.6, we can clearly see that the average performance of PRF models is superior to

Table 3.4: PRoc compares with BM25+Rocchio and LM+RM3 on WT2G. The percentages in the parenthesis are the improvement gains over the classic Rocchio's model and RM3. A "*" indicates a statistically significant improvement over the classic Rocchio's model baseline, and a "+" indicates a statistically significant improvement over the RM3 model baseline according to the Wilcoxon matched-pairs signed-ranks test at the 0.05 level. The bold phase style means that it is the best result.

| # of feeback terms | PRoc1 | PRoc2 | PRoc3 | BM25 + Rocchio | LM + RM3 |
|---|---|---|---|---|---|
| 10 | **0.3415**$^{*+}$ | **0.3415**$^{*+}$ | 0.3385$^{*+}$ | 0.3091 | 0.3212 |
| 20 | **0.3501**$^{*+}$ | 0.3403$^{*+}$ | 0.3424$^{*+}$ | 0.311 | 0.3225 |
| 30 | 0.3452$^{*+}$ | 0.3456$^{*+}$ | **0.3461**$^{*+}$ | 0.3082 | 0.3255 |
| 50 | 0.3513$^{*+}$ | 0.3406$^{*+}$ | **0.3525**$^{*+}$ | 0.3162 | 0.3242 |
| Average | 0.3470 (11.54%, 7.30%,) | 0.3420 (9.93%, 5.75%,) | 0.3449 (10.86%, 6.65%,) | 0.3111 | 0.3234 |

Table 3.5: PRoc compares with BM25+Rocchio and LM+RM3 on WT10G. The percentages in the parenthesis are the improvement gains over the classic Rocchio's model and RM3. A "*" indicates a statistically significant improvement over the classic Rocchio's model baseline, and a "+" indicates a statistically significant improvement over the RM3 model baseline according to the Wilcoxon matched-pairs signed-ranks test at the 0.05 level. The bold phase style means that it is the best result.

| # of feeback terms | PRoc1 | PRoc2 | PRoc3 | BM25 + Rocchio | LM + RM3 |
|---|---|---|---|---|---|
| 10 | 0.2290$^{*+}$ | 0.2267$^{*+}$ | **0.2308**$^{*+}$ | 0.2143 | 0.2098 |
| 20 | 0.2272$^{*+}$ | 0.2219$^{*+}$ | **0.2287**$^{*+}$ | 0.2147 | 0.2168 |
| 30 | 0.2260$^{*+}$ | **0.2264**$^{*+}$ | 0.2261$^{*+}$ | 0.2084 | 0.2151 |
| 50 | 0.2202$^{*+}$ | 0.2206$^{*+}$ | **0.2245**$^{*+}$ | 0.2039 | 0.2136 |
| Average | 0.2256 (7.28%, 5.52%,) | 0.2239 (6.47%, 4.72%,) | 0.2275 (8.18%, 6.41%,) | 0.2103 | 0.2138 |

the basic models in most cases. The classic Rocchio's model achieves improvements of 13.31%, -0.41%, 2.34% and 4.22% over BM25 on the Disk4&5, WT2G, WT10G and GOV2 collections, while RM3 obtains significant improvements over LM (4.05%,

Table 3.6: PRoc compares with BM25+Rocchio and LM+RM3 on GOV2. The percentages in the parenthesis are the improvement gains over the classic Rocchio's model and RM3. A "*" indicates a statistically significant improvement over the classic Rocchio's model baseline, and a "+" indicates a statistically significant improvement over the RM3 model baseline according to the Wilcoxon matched-pairs signed-ranks test at the 0.05 level. The bold phase style means that it is the best result.

| # of feeback terms | PRoc1 | PRoc2 | PRoc3 | BM25 + Rocchio | LM + RM3 |
|---|---|---|---|---|---|
| 10 | 0.3271$^{*+}$ | **0.3294**$^{*+}$ | 0.3288$^{*+}$ | 0.3126 | 0.3071 |
| 20 | 0.3303$^{*+}$ | **0.3325**$^{*+}$ | 0.3315$^{*+}$ | 0.3164 | 0.3141 |
| 30 | **0.3323**$^{*+}$ | 0.3316$^{*+}$ | 0.3320$^{*+}$ | 0.3175 | 0.3167 |
| 50 | 0.3242$^{*+}$ | **0.3313**$^{*+}$ | **0.3313**$^{*+}$ | 0.3181 | 0.3183 |
| Average | 0.3285 (3.89%, 4.58%,) | 0.3312 (4.74%, 5.44%,) | 0.3309 (4.65%, 5.35%,) | 0.3162 | 0.3141 |

7.98%, 3.64% and 3.32%) on all the four collections[7]. The effectiveness of pseudo relevance feedback is re-confirmed in this set of experiments. The classic Rocchio's model, fails to obtain improvement on the WT2G collection. This indicates that the Rocchio's model is not so robust as RM3 in this case. However, the Rocchio's model outperforms RM3 on the Disk4&5 collection significantly while RM3 performs better than the classic Rocchio's model on the WT2G collection. On the WT10G and GOV2 collections, their results are very close. Therefore, the Rocchio's model is generally comparable to RM3 so that it is still competitive to be a strong baseline.

In general, the performance of our proposed PRoc models is close on all the four collections, and all of them obtain more improvements over the basic models than

[7]The computation of these percentages is based on the average performance of the Rocchio's model and RM3 in Table 3$\sim$6 and MAPs in Table 2.

the Rocchio's model and RM3. Specifically, from Table 3.3 to Table 3.6, we observe that all the three proximity-based Rocchio's models outperform the classic Rocchio's model (2.00% - 11.54%) and state-of-the-art RM3 (4.78% - 12.75%) significantly on all the four collections. The results demonstrate the effectiveness of the three PRoc models. Although the measures in our PRoc models are different, all of them can successfully model the proximity information to some extent. Furthermore, the PRoc models perform more robustly than the classic Rocchio's model. It indicates that proximity plays an important role in discriminating relevant expansion terms from irrelevant ones.

In addition, from Table 3.4 we observe that PRoc3 outperforms the other two on the WT2G collection. On the other three collections, the performance of all the three PRoc models is very close. Generally, PRoc3 is slightly more effective than the other two PRoc models.

### 3.3.3 Effectiveness of Window Size

In our proposed PRoc models, there are two important parameters: (1) $\beta$ in the feedback models controlling how much we rely on the original query and the feedback information and (2) window size parameter $wSize$ in the calculation of the proximity-based frequency. In our preliminary experiments, we observed that the influence of

$\beta$ is similar to that in [YHHL10a], which investigated this parameter thoroughly. Since we mainly focus on the study of proximity evidence, detailed discussion about $\beta$ will not be made in this work.

$wSize$ is a key parameter for most proximity measures because it determines the distance in which terms are considered to be related. Thus, how to find an appropriate window size is very important for adapting the proximity measures. In this section, we attempt to discover some useful evidence for obtaining optimal $wSize$ values. Particularly, $\sigma$ in PRoc2 is also interpreted as the window size.

From Figure 1 to 4, we show how the performance of our PRoc models changes with $wSize$ on different collections. We investigate a large range of $wSize$ from 10 to 1500, and the numbers of expansion terms are 10, 20, 30 and 50. Generally, the values of $wSize$ affect the performance of all the PRoc models extensively. In the second subfigure of Figure 4, the best MAP is 0.3205 when the number of expansion terms is 50, and it falls to 0.2286 when $wSize$ is 1500. Almost 30% of performance is lost in this case.

For PRoc1, PRoc2 and PRoc3, although they are based on different measures, their curves fluctuate similarly on the same collection with various numbers of expansion terms. In contrast, the curves of each PRoc model are various extensively on different collections. This demonstrates that the influence of $wSize$ is collection-based. However, the best $wSize$ values for PRoc models are not the same, not even close to each other. For example, on the disk4&5 collection, optimal $wSize$ values for PRoc1 are 80, 50, 100 and 80 over 10, 20, 30 and 50 expansion terms, and the corresponding values for PRoc1 on WT10G is 100, 30, 10 and 10. Thus, the optimal values of $wSize$ depend on the proximity measures and the collections.

Another phenomenon is that the more the expansion terms are selected, the more the performance is affected by $wSize$. Normally, the performance of PRoc models drops when $wSize$ takes a relatively large value. However, to what extent the performance is affected is determined by the number of expansion terms. Specifically, in Figure 2, while $wSize$ is relatively small, the performance of PRoc models with 50 expansion terms is the best. However, when $wSize$ is larger than 200, the curves for 50 expansion terms are constantly below all the others. As an additional example, when the number of expansion terms is 30, the performance of PRoc models is the second worst in most cases when $wSize$ is 1500.

This is reasonable because the accumulated influence of proximity information for 50 expansion terms is larger than that of the small numbers. When $wSize$ increases, it is very likely that more noise is adopted in the expansion term selection. The more expansion terms are there, the more noisy information is involved. Thus, the $wSize$ must be set very carefully when the number of expansion is larger than 30 in our

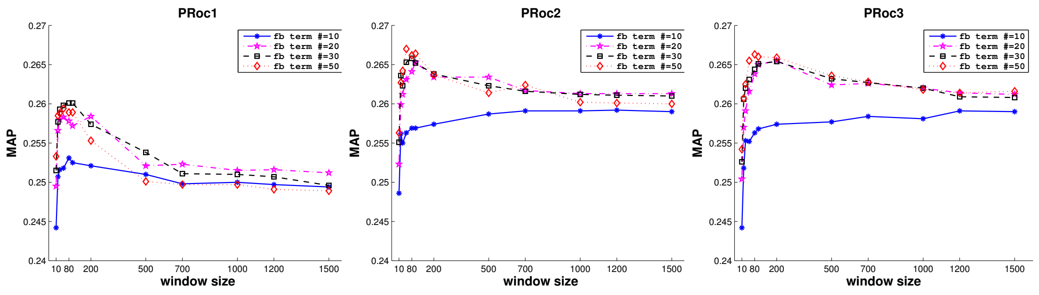


Figure 3.1: PRoc1, PRoc2 and PRoc3 over disk4&5 with 10, 20, 30 and 50 expansion terms

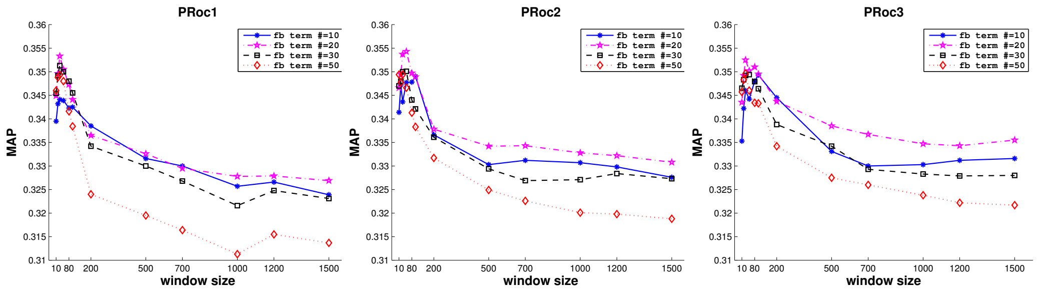


Figure 3.2: PRoc1, PRoc2 and PRoc3 over WT2G with 10, 20, 30 and 50 expansion terms

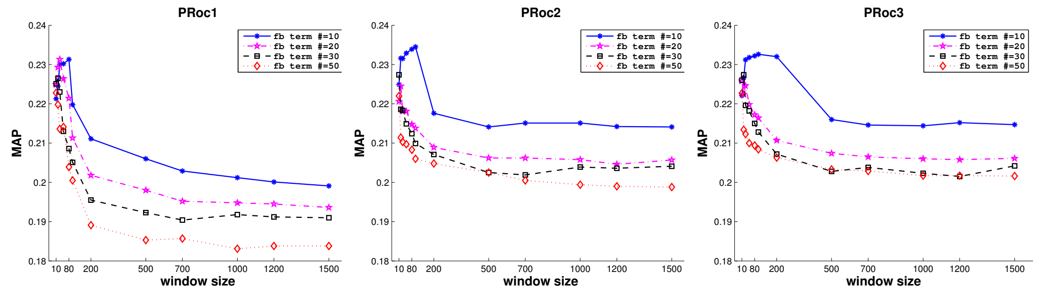


Figure 3.3: PRoc1, PRoc2 and PRoc3 over WT10G with 10, 20, 30 and 50 expansion terms

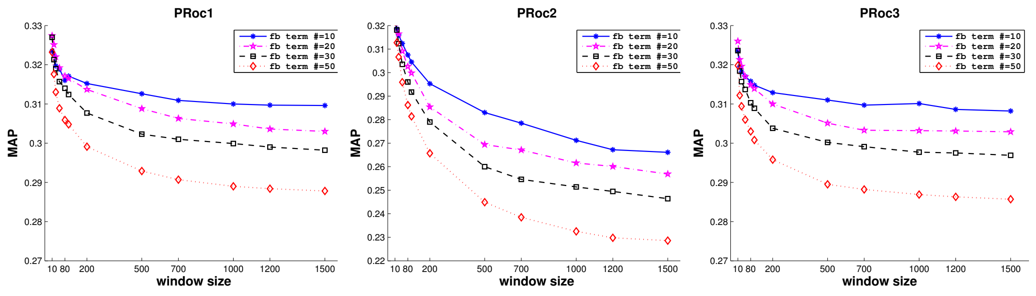


Figure 3.4: PRoc1, PRoc2 and PRoc3 over GOV2 with 10, 20, 30 and 50 expansion terms

case.

Additionally, the influence of $wSize$ on PRoc1 is more significant than on the other two over WT10G. Meanwhile, $wSize$ affects PRoc2 more significantly over GOV2 than PRoc1 and PRoc3. Overall, the PRoc3 model is less sensitive than the other two PRoc models according to our experiments.

In summary, a big challenge is to find an optimal value since the value space is very large without any constraints. It is very time-consuming to try every possible value in the relevance feedback process. To narrow the value space of $wSize$, we attempt to find a rule to direct the searching of optimal values. Based on our experimental results, we conjecture that there are two factors affecting the choice of $wSize$: the average document length (ADL) and the size of a collection. Intuitively, if the average document length is large, it is more likely to have more than one topic in a document which leads to involving more irrelevant terms. Besides, as the increase of the size of a collection, it is more likely to bring irrelevant documents into the feedback process.

In order to minimize the negative influence of noise, the values of $wSize$ should be relatively small especially when the ADL or collection size is large. Only the closest terms will be considered to avoid the selection of irrelevant terms. In our experiments, this rule is supported by some evidence. The ADL of Disk4&5 is 334 and there are only 528,155 documents in this collection. The performance of all the PRoc models is not affected by $wSize$ so extensively as that on the other collections. When there are only ten expansion terms, the optimal value of $wSize$ can be as large as 1500. On WT10G, which has 1,692,096 documents and an average document length of 426, the optimal $wSize$ values are greater than 50 but smaller than 200. For WT2G, even it has fewer documents (247,491) than other collections, the optimal $wSize$ values are in a range of (30, 50) because of its long ADL (722). GOV2 is the largest collection of 25,178,548 documents in our experiments, and its ADL(679) is only smaller than that of WT2G. As a result, the optimal values of $wSize$ for GOV2 is the smallest one. It is always 10 in our case. In summary, we can use this rule to narrow the search space of optimal $wSize$ values. If a collection has plenty of documents or its ADL is large (e.g., more than 700), it is always good for us to start from 20 or smaller. Otherwise, we can try a larger starting value like 50 or more.

### 3.3.4 Comparison with PRM

We also compare our proposed model with the recently developed position relevance model (PRM) [LZ10], which is an extension of the relevance model. In particular, PRM takes into account term positions and proximity with the intuition that words closer to query words are more likely to be related to the query topic, and assigns more weights to candidate expansion terms closer to the query. To make the comparison fair, we train our parameters on the Terabyte05 topics and use Terabyte06 [8] topics on the GOV2 collection for testing as Lv. *et al.* did in [LZ10]. Since we do not give results for the Million Query Track so far, we do not compare our method with PRM on the ClueWeb collection with the topics of this track. In [LZ10], parameter $\mu$ in the Dirichlet smoothing is set to an optimal value of 1500, and we set $b$ in our basic model, BM25, empirically to 0.3 [ZHH11]. As we mentioned previously, the performance of BM25 and LM with Dirichlet smoothing does not differ significantly on the GOV2 collection. Therefore, this setting will not affect the comparison. Since PRoc3 is the most robust and performs the best generally, it is selected to make this comparison. There are two versions of PRM, PRM1 and PRM2. The results of

[8]http://trec.nist.gov/data/terabyte.html

Table 3.7: PRoc compares with the classic Rocchio's model, RM3, PRM1 and PRM2 on Tera06 dataset. The bold phase style means that it is the best result.

| | PRoc3 | Rocchio | RM3 | PRM1 | PRM2 |
|---|---|---|---|---|---|
| MAP | 0.3283 | 0.3156 | 0.3131 | **0.3322** | 0.3319 |
| P@10 | **0.5800** | **0.5800** | 0.5043 | 0.5306 | 0.5490 |
| P@30 | **0.5260** | 0.5167 | 0.4660 | 0.4884 | 0.4871 |
| P@100 | **0.3756** | 0.3664 | 0.3576 | 0.3671 | 0.3741 |

RM3, PRM1 and PRM2 are directly from [LZ10].

In Table 3.7, PRoc3 outperforms the classic Rocchio's model and RM3 significantly in terms of the MAP metric, and it is only slightly inferior to PRM1 and PRM2 by 1.19% and 1.1% respectively. On the P@10, P@30 and P@100 metrics, PRoc3 obtains the best results over all the other four models and outperforms RM3, PRM1 and PRM2 significantly. All these significant tests are based on the Wilcoxon matched-pairs signed-ranks test at the 0.05 level. This shows that the retrieval accuracy of our proposed model is better than that of the PRF models in the language modeling framework in this case. Since the results of PRM1 and PRM2 are optimized, it is reasonable to state that our propose model is at least comparable to the most recent progress.

# Chapter 4

# TopPRF: A Probabilistic Framework for Integrating Topic Space into Pseudo Relevance Feedback

## 4.1 The main challenge of applying topics for PRF

Traditionally in classic PRF models like Rocchio [Roc71a] or RM3 [LC01b], all the top $k$ feedback documents are assumed to be equally relevant. The weights of candidate feedback terms in them are calculated based on their features only. Once the documents are chosen, their reliability[9] is not considered anymore. Generally, terms in different feedback documents with the same weight (e.g., tf-idf score) are considered to be equally reliable for query expansion. According to our preliminary experiments, we use BM25 [RWJ$^+$94b] with optimal parameters to investigate how reliable the top $k$ feedback document are. As we know, BM25 is one of the most popular models and has been widely used as the basic model of probabilistic PRF [HHW$^+$06b, CKC$^+$08, RZ09, MHY12]. Surprisingly, when we assume these documents are relevant (that is why the process is called pseudo relevance feedback

[9]In this paper, "reliability" of a feedback document refers to how reliable it is to be relevant.

because the documents are not identified by human efforts), the ratio of really relevant documents in top $k$ is not high. On the WT10G dataset with TREC2001 queries, approximately 1/3 in top 3, 3/5 in top 10, 2/3 in top 30 and 4/5 in top 50 documents are irrelevant. Those irrelevant documents import noisy information which can harm the overall performance of PRF significantly. Meanwhile, even a document is relevant, it can also contain irrelevant contents. Terms in these irrelevant contents will influence query expansion negatively as well.

In a document, a relevant term is surrounded by other terms which can be either relevant or not. Without extra information, it is hard to identify the relevance of a term from terms around it, especially when the document itself can be irrelevant. Recently, researchers begin to apply topic models [SK13, WZWS12, YA09, YA08, YHL11b] for PRF to solve this problem. They attempted to find feedback terms in the most relevant topic(s) and expanded the original query with them. In other words, they used relevant topics to replace feedback documents for PRF. The advantage of this kind of method is breaking the constraint of document scope. Because topic modeling considers the co-occurrence of terms within the whole collection for training, term relations can be conducted. For example, when a term $t_1$ always appears with query terms, it is very likely to be relevant to the original query and has a high probability in the query related topic $K$. If there is another term $t_2$ co-occurs with $t_1$ in other documents, $t_2$ will have a high probability in $K$, too. It is possible that $t_2$ does not appear with query terms many times because query terms can have synonyms. In this case, we can find $t_2$ through topical information and expect the top terms in the relevant topics are all relevant, too.

There is a big obstacle for this application which we call a "Fuzzy Topic" [10] problem. In this paper, a topic is defined as the main theme or subject contained in a (set of) document(s), which can be represented by a list of terms with the corresponding probabilities of generating the terms from the topic. A topic can be considered as a particular distribution of terms in vocabularies. It is not a very clear concept even for human beings. In other words, topics are abstract. Hence, it is difficult to identify how many and what topics a document is really about. For example, one may consider a document is about the "finance" topic. A different person may think the document contains two topics "stock" and "bond", and individuals who regard the document as "investment" related can also be correct.

We can think about this problem more generally. Suppose we have a corpus and define the information as global information. The global information is fixed if the corpus is not changed. When we decide to discover how many topics in this corpus, we attempt to split the global information into sub-information pieces and make topics

[10] "Fuzzy" may not be the best work this problem because it was used by some other terminologies in computer science. Here we use it just to emphasize the unstable of topics

through them. Topics are the aggregations or the segments of these sub-information pieces. Since information itself is hard to be quantified and people can interpret it differently, there are reasonably many ways to organize these sub-information pieces and generate different topics. So the "fuzzy topic" problem appears naturally and the best number of topics in a corpus cannot be determined.

Previous work expected to find relevant terms in particular topics. So the problem of selecting relevant terms changes to identify relevant topics. However, as we can see from the above example, the information in each topic can be divided or aggregated. This depends on how many topics we assume to be there. If we attempt to obtain particular topics through some rules, the desired information on these topics can be quite different when the topic number is changed. Unfortunately, when we use popular topic models to discover topics from a corpus, there is not an appropriate way to determine the topic number. Previous researchers proposed some methods to find an optimal topic number [BNJ03b, GS04, GT04]. They attempted to optimize the parameter by measuring which value can improve the final performance of a particular target (e.g., classification) the most, but none of them is IR or PRF oriented. As a result, the performance of the applications based on these topics will be very unstable, and the unstableness will propagate to these methods and affect the overall performance. This is why topics are coarse to some extent [WC06] and hard to be applied in IR. More evidence will be shown in the Section 4.3 to demonstrate how topics change significantly with different topic numbers.

Besides the challenge of identifying topics, we think another latent problem is the loss of topical information. Using a few topics can neglect useful information in other ones, even when they are relevant to the query topic. For example, terms with higher probabilities in the selected topic(s) are considered to be more important for PRF. However, they can also appear frequently in other topics and are not so informative. In that case, terms which have even probabilities in many topics should be less important. Without the information of full topics, we will miss this kind of features.

Generally, if we decide to utilize particular topics, we have to decide or seek an optimal topic number first. Then the effect of "fuzzy topic" problem is inevitable due to significant change of topics and the loss of topical information.

Instead of identifying relevant feedback terms directly, our idea is to select feedback terms based on the relevancy of feedback documents using topical information. For example, [HMH13] applied a machine learning technique co-training to label feedback documents as relevant/irrelevant and [YHM12b] took the original score of feedback documents into account. Previous work started to consider whether a feedback document is really relevant or not, but was not from the view of topics. In order to address the "fuzzy topic" issue when using topical information, we propose a new concept of "topic space" in the next section.

## 4.2 Topic Space

To identify whether a feedback document is relevant to the query, an intuitive way is to measure the similarity between feedback documents and really relevant documents. Since documents are represented by terms, traditional similarity measurements are also based on terms, e.g., vector space model or cosine similarity. In this work, we use topics to represent a document. Topics contain more general information than terms because the former is a distribution of all terms in the vocabulary. By using topics to represent a document can we discover associations at a different level. To some extent, topics can reveal more semantic information than terms. So it is worth considering about how to utilize this to evaluate feedback documents.

To effectively apply topical information for PRF and reduce the influence of fuzzy topics, we decide to take a complete view of the documents and propose a new concept "topic space". A document is considered as a mixture of different information. If we treat a document as a point in the space, we can use different coordinate systems to describe it and locate it. When we change the dimension and the meaning of each axis of the system, we will have a different view on the document. In the vector space model [SWY75a], the dimension of the system is the size of the vocabulary, and each dimension is the weight of a particular term given a document. The projection of the document point on each dimension demonstrates how important a term is in this document. What will we obtain if we create a coordinate system based on topics to describe a document?

When integrating topic modeling on a corpus, suppose we set the number of topics to be $M$ and then we will have a set of topics $z_1, z_2 ... z_M$. If a document $d$ is about $z_1$ and $z_2$, the probabilities $p(z_1|d)$, $p(z_2|d)$ should be obviously higher than $p(z_i|d)$ while $i \neq 1$ *or* 2. We build a coordinate system [SYY98] based on the topics we obtain. The coordinates ($p(z_1|d)$, $p(z_2|d)$, ... $p(z_M|d)$) are used to denote document $d$ and the summation of $p(z_i|d)$ is 1 for $i \in \{1, 2...M\}$. We define the coordinate system as a *topic space* and documents are vectors in this space. So the system has $M$ dimensions, and each dimension denotes the conditional probability of a topic given a document. When we change $M$, we change the way to describe the document point, or the mixture of information in other words. No matter how we change the system, the document itself is unchanged in the topic space. In this case, we can always precisely describe the document with all the topic coordinate information. Unlike the term-based coordinate system, the matching of different documents/queries can be done beyond bag-of-words techniques in the topic space.

With this new concept, it is simple to map a document into space and apply sophisticated space-related methods. For instance, we can define a *topic vector* as starting from the original point and ending at the document point, and then methods applied in the vector space model [SWY75a] can be used as well. An example is shown in Figure 4.1. Three documents are represented as three vectors when there are three topics. If only some of the topics are used, we just investigate projections on some dimensions of the document, which is not complete and cannot describe the document accurately. When we attempt to discover useful information among documents, we always use the full topic coordinates to avoid biased topical information. We will show some experimental results in Section 4.7.1 and 4.7.2 to justify this.

To measure the reliability of each document and choose terms to expand the original query on the evaluation, we do not focus on a particular topic. Instead, we use the coordinates of a document in the topic space to implement our ideas. To this end, we have two following assumptions.

**Assumption 1** *If two documents are similar on the topical level, their positions in the topic space will be close even when the number of topics is changed.*

For instance, if two documents $d_1$ and $d_2$ are about "stock investment" with 10 topics $z_1, z_2 ... z_{10}$. If both $z_3$ and $z_5$ are related to "stock investment", $p(z_3|d_1)$, $p(z_5|d_1)$, $p(z_3|d_2)$ and $p(z_5|d_2)$ should be obviously higher than other conditional

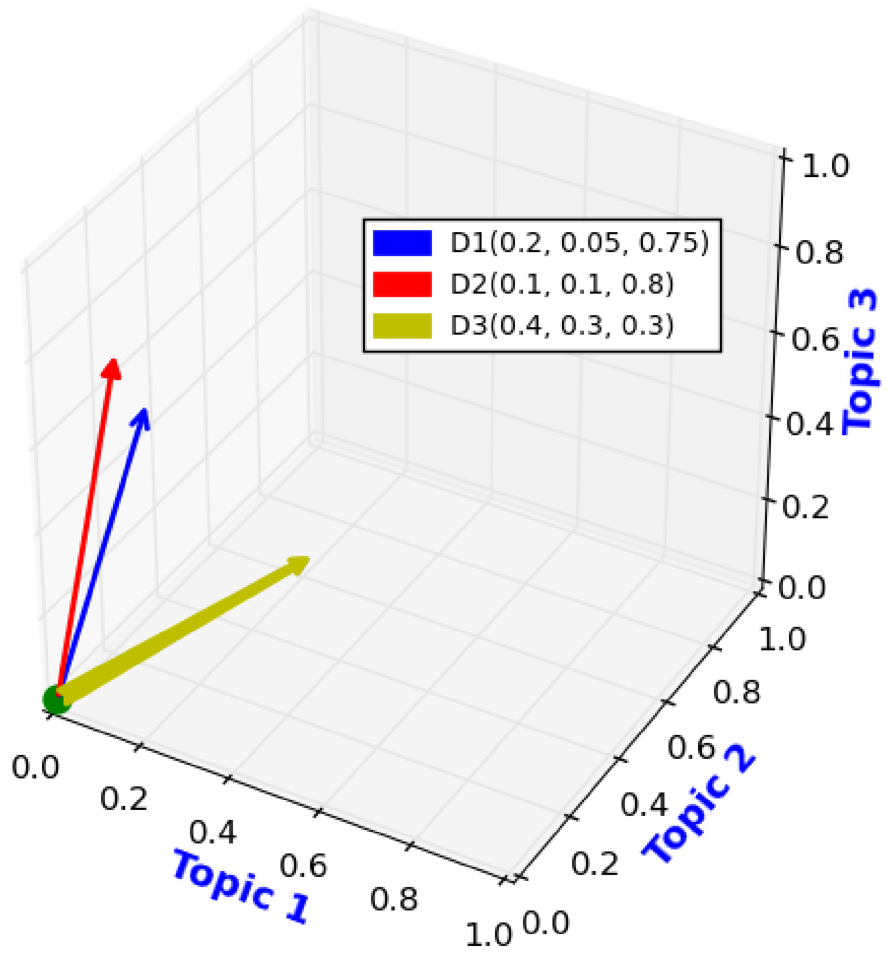


Figure 4.1: Documents represented in a 3-topic space. Numbers in brackets are the coordinate values for each document.

probabilities. If we change the topic number to 5, there may be only one topic $z_2$ related to “stock investment”. The two documents should still have similar topic coordinates while their contents are very similar. However, suppose $d_1$ is a query and we still set topic number to 10. If we decide to choose one topic for selecting feedback terms, no matter how we choose $z_3$ or $z_5$, information in the neglected topic will be lost.

An issue we concern is how to measure the similarity/closeness of two documents. Here we propose two models. Our purpose is to research whether the topic similarity between documents can help improve PRF instead of which similarity method will be the best. Therefore, we intuitively choose two very popular similarity measurements. Other similarity methods will be studied in the future work.

Firstly, we can consider the cosine similarity between the topic vectors of two documents. The association between topic vectors should be more stable than particular topics while we view the documents in the complete scope of topics. Sometimes, it is also a useful sign when two documents both have low probabilities on a particular topic. Secondly, if we use the distance between two document points to measure their similarity, there will be plenty of candidate formulas to investigate, e.g. Euclidean distance. In this paper, we will propose two methods named TS-COS and TS-EC to apply topic-similarity scores for estimating the reliability of a feedback document. Details of these two methods will be presented in Section 4.4.3 and 4.4.4. The higher score it has, the more likely it is relevant. The scores of these documents will affect the weights of candidate feedback terms in them. Terms in documents of high weights are considered to have more impact for query expansion. In Figure 4.1, coordinates for D1 and D2 are (0.2, 0.05, 0.75) and (0.1, 0.1, 0.8) respectively. Because both of them have a large portion of topic 3, they are very close in the topic space as shown in Figure 4.1. That is the feature we decide to make use of. In this paper, we apply the cosine similarity and Euclidean distance to measure the closeness of two documents in the topic space. In addition, in order to apply the two methods and obtain the weights of each document, we select a small group of feedback document as samples, measure the average similarity score of each feedback documents to these samples as its weight. Details about how and why to choose the samples are introduced in Section 4.4.3.

**Assumption 2** *In PRF, the feedback documents are considered to be relevant. The fewer topics a document contains, the lower risky the document is.*

When we assume the top $k$ feedback documents in PRF are relevant, if a document is only about one topic, i.e., one topic has a much larger probability than others given the document, we can consider all the contents of the document are relevant, or we denote it a “pure” document. Otherwise, if the topic distribution of a document is very even, it is reasonable to think some parts of the document are not relevant. Thus, it is risky to import terms from them. In that case, a less pure document is not so reliable when evaluating the weights of candidate feedback terms. Inspired by the traditional information theory, we measure the purity of topical information in a document through "entropy" by replacing the probabilities of terms within those of topics. We also propose a TS-Entropy method on this assumption to address the negative effect of partially relevant documents. Details of the TS-Entropy method will be presented in Section 4.4.5.

To the best of our knowledge, our proposed approaches are novel for integrating all topical information instead of selected topics in PRF under the probabilistic framework. A document is represented as a mixture of all topics, and the latent topical information is retained to represent the meaning beyond individual terms. According to our Assumption 1 and experimental results presented in Section 4.6 and 4.7, our proposed methods are not sensitive regarding retrieval performance to the settings of topic numbers.

## 4.3 Preliminary Study of "Fuzzy Topic" in Pseudo Relevance Feedback

Before incorporating topic space in PRF, we study the characteristics of latent topics generated by LDA in this section. Section 4.3.1 analyzes how the terms in the topics change according to selections of topic numbers, and how we plan to deal with such issues in PRF. In Section 4.3.2, we discuss why topic space can help PRF.

### 4.3.1 Observations of Fuzzy Topic

We will show how fuzzy topics are obtained when changing the topic number for a very popular topic model, LDA [BNJ03b]. To show how a topic changes with the topic number, we choose the topic which is most likely to generate the query. This is the traditional way used in previous studies [AB11, YHL11b, SK13]. Then we check how it changes with different topic numbers. All the experiments presented in this section are done on the TREC GOV2 collection with official queries[11]. We still use BM25 [RWJ$^+$94b], a classic probabilistic model, as the first-pass retrieval model. Top 30 documents are chosen to train the LDA model. The topic number is set to

[11]In this paper, in order to avoid confusion, "topic" only refers to topics obtained through topic modeling. We do not use "topic" defined by NIST to indicate queries for all TREC datasets as some previous research does. Instead, we use the term "query".

Table 4.1: Topics given topic# 5, 10 and 30 on query 802 "Volcano eruptions global temperature" and query 804 "ban on human cloning"

| Term Ranks | Volcano eruptions global temperature | | | ban on human cloning | | |
|---|---|---|---|---|---|---|
| | 5 | 10 | 30 | 5 | 10 | 30 |
| 1 | volcano | volcano | volcano | clone | clone | clone |
| 2 | nbsp | can | can | human | human | human |
| 3 | erup | volcan | flow | embryo | research | ban |
| 4 | volcan | flow | activ | ban | embryo | research |
| 5 | magma | activ | mount | research | us | embryo |
| 6 | can | erup | hazard | us | ban | us |
| 7 | flow | earthquak | **ash** | cell | cell | cell |
| 8 | activ | mount | magma | quot | will | state |
| 9 | **ash** | usg | water | will | produc | will |
| 10 | earthquak | magma | pyroclast | reproduct | purpos | creat |
| 11 | mount | hazard | scientist | state | creat | produc |
| 12 | gase | scientist | gase | mai | moral | purpos |
| 13 | second | rock | erupt | creat | onli | new |
| 14 | rock | lava | lava | stem | stem | legisl |
| 15 | usg | gase | peopl | therapeut | prohibit | onli |
| 16 | gas | erupt | state | be | mai | be |
| 17 | hazard | pyroclast | hot | purpos | be | prohibit |
| 18 | mb | peopl | lahar | onli | who | reproduct |
| 19 | lava | water | like | who | new | allow |
| 20 | monitor | caus | mile | act | transfer | who |
| 21 | water | hot | chang | moral | act | mai |
| 22 | 1 | dioxid | near | new | allow | need |
| 23 | erupt | debri | rock | ethic | need | stem |
| 24 | 0 | area | hawaii | prohibit | ethic | feder |
| 25 | dioxid | lahar | type | transfer | medic | act |
| 26 | scientist | mile | move | allow | legisl | life |
| 27 | temperatur | occur | includ | creation | feder | prohibi |
| 28 | degre | chang | st | legisl | attempt | medic |
| 29 | caus | st | ground | reason | wai | develop |
| 30 | pyroclast | monitor | earth | believ | time | time |
| 31 | measur | helen | cascad | scientif | issu | attempt |
| 32 | vent | like | caus | medic | call | issu |
| 33 | state | state | observatori | hous | practic | transfer |
| 34 | pressur | alaska | debri | genet | effect | support |
| 35 | geolog | ground | mudflow | prohibi | embryon | 1 |
| 36 | alaska | **ash** | cloud | attempt | state | congress |
| 37 | hot | move | system | life | life | creation |
| 38 | includ | peak | sulfur | support | now | call |
| 39 | peopl | cascad | form | like | creation | effect |
| 40 | debri | increas | locat | technolog | scientist | first |
| 41 | st | includ | danger | work | support | ethic |
| 42 | lahar | hawaii | aircraft | year | like | law |
| 43 | move | movi | onli | question | peopl | practic |
| 44 | ground | type | helen | individu | egg | technolog |
| 45 | helen | near | thousand | effect | prevent | reproduc |
| 46 | hawaii | geolog | hundr | reproduc | 2 | now |
| 47 | mai | continu | washington | embryon | reason | requir |
| 48 | area | carbon | call | wai | requir | genet |
| 49 | occur | produc | surfac | benefit | reproduc | 2 |
| 50 | type | system | crater | requir | gener | live |

5, 10 and 30, and we show the top 50 terms ranked by their probabilities given the topic. All terms are processed by Porter's Stemmer [Por80]. We arbitrarily choose two queries: 802 and 804 as examples and fix parameters except the topic number for all the experiments. So the top 30 documents are the same for LDA.

Although all the topics we obtain are assumed to be the "most relevant", they are different. If we attempt to choose feedback terms from them, we will be somewhat confused due to ranks of these terms. As we can see from Table 4.1, when the topic number changes, the term lists are different. Usually, 10 to 50 terms will be chosen as feedback terms for query expansion. However, when we just increase the topic number from 5 to 10, even the list of top 10 terms in the table changes significantly. For example, when the topic number is 10, "ash" is ranked 36 in the topic of query 802. But when the topic number is set to 5 and 30, its rank raises to 10 and 7. How can we evaluate the relevance of "ash" in this case?

If we change our view from the "terms" in these topics to "topics" themselves, there is something different. As we consider, a topic can represent a particular kind of information. When we change the topic number, the information can be divided into 2 or 3 parts or be combined with others. The similarity of two documents on a particular topic will be reflected more or less in the new coordinate system and identify their closeness in the topic space. Like the example given in Section 4.2, when two documents are both mainly about one topic, they will also have its subtopics when the topic is split into two parts. Meanwhile, we use the full topic coordinates of documents to measure the reliability and adapt them into the traditional term-based framework as an enhancement. In Section 4.7.1, we will show that the ranks of documents are more stable when the topic number changes. Compared to the conditional probability of a topic given the document, the internal changes of terms in this topic are more frequent and significant. Therefore, for these “fuzzy topics”, it is hard to tell which term should be more relevant than the other while their relative ranks are not stable. If a researcher attempts to choose a term based on the rank or probability information on a given topic, it is very difficult for him/her to make a choice.

### 4.3.2 Why Topic Space Can Help PRF

Traditional term-based PRF approaches choose feedback terms from top feedback documents and do not consider whether these documents are entirely relevant, partially relevant or not relevant to the query. To solve this problem, researchers have used topic modeling to extract the topics from text collections and choose feedback terms from the particular topic(s). Different strategies have been used to identify the topic(s) related to the original query, and terms appear with higher probabilities in the topic(s) are regarded to be more semantically related to the original query. The advantage of this type of methods is breaking down the constraint of document scope. Because topic modeling considers the co-occurrence of terms within the whole collection for training, and the relation among terms across documents can be considered. For example, if a term $t_1$ always appears frequently with given query terms, $t_1$ is highly likely to be relevant to the original query and has a high probability in the query related topics. If there is another term $t_2$ co-occurs with $t_1$ in other documents, $t_2$ will have a high probability in these query related topics, too. It is possible that $t_2$ does not co-occur with the query terms frequently because query terms can have synonyms. In this case, $t_2$ is also a good feedback term but can not be identified by traditional term-based PRF approaches. Using topical information, $t_2$ can be identified and utilized in PRF. However, the topic modeling approaches have randomness and generate different groups of topics if performed multiple times.

Therefore, we investigate this "fuzzy topic" issue and plan to find a more robust way to deal with this issue. We do not focus on identifying particular topics. Instead, we use topical information to evaluate the quality of feedback documents rather than selecting feedback terms. That is an alternative way to improve PRF. So we propose the "topic space" which uses topics as coordinates and represents feedback documents as topic vectors. The idea is inspired by the vector space model [SWY75b] for the term-based document representation. One advantage is that we can apply similarity methods for documents on the topical level. Different from traditional term-based methods, topical information can help to discover the latent semantic similarity among documents. Our proposed idea can be a good complementary for the current term-based PRF methods like Rocchio. Another advantage is that we do not deal with the dilemma of choosing topics. Our ultimate goal is to evaluate the relevancy between documents and the query. Therefore, we can address the fuzzy topic issue by using all topics to represent documents. In the next section, we will propose a new probabilistic framework called TopPRF and present how to integrate the topic space into PRF naturally instead of dimension reduction.

## 4.4 Integrating Topic Space into Pseudo Relevance Feedback

LDA is widely used in the text mining area [WB11, TJZ08, MSZ07, GS04, YA09], and the retrieval performance using LDA is even better than another popular model PAM in PRF [YA09]. We, therefore apply LDA to obtain latent topics for our proposed methods. In the rest of this section, we will introduce how to build the topic space via the LDA model, explain how we use topics for PRF and propose three methods to select good feedback terms in details.

Another issue we consider is whether we should apply LDA on the whole collection. Building topics on the whole corpora will take a large amount of time. When the size of the collection is very big, it is even harder to determine the number of topics. It should be a large number because we have millions of documents. Because of the "fuzzy topic" problem, we are not able to determine the best topic number. Besides, it can be different for various queries. Although offline topic modeling may save time for further usage, this problem cannot be avoided. Besides, even we do an offline LDA on a medium dataset like WT10G which contains 1,692,096 documents, it is reasonable to set a large topic number like 5,000. Otherwise, the topics we obtain will be very general in information. When we compare two documents, most features in their topic vectors will be close to 0, i.e., their topic vectors are very sparse. In this case, their similarities will always be near 1 when we calculate similarity scores. Consequently, documents cannot be identified through topical information. Experimental results reported in [YA08] confirmed that topics discovered on the whole corpora are too coarse-grained for query expansion. In Section 4.6.4, we will show some experimental results over 100 and 200 topics which are obtained through offline LDA on the whole corpus. While the resource cost is much higher than our online strategy on the top documents, the performance is not improved. So in this paper, we only use the top $k$ feedback documents as the source of LDA.

In this section, we will first introduce the LDA model which will be used for generating topic space, and then present how we integrate the topic space into the new framework TopPRF. Based on this framework, we will describe how we measure the reliability of feedback documents through three newly proposed models: TS-COS, TS-EU and TS-Entropy. These three models are built according to our assumptions in Section 4.2. TS-COS and TS-EU are two similarity-based models designed for Assumption 1 and TS-Entropy model is designed to measure the purity of topical information in a document for Assumption 2.

### 4.4.1 Generating Topic Space via LDA

The Latent Dirichlet Allocation (LDA) model [BNJ03b] deals with the problem of modeling text corpora and other collections of discrete data. LDA is applied to find short descriptions for documents of a collection and enable discovering and preserving essential statistical relationships that are useful for classification, document diversify, summarization, similarity and relevance judgments.

When modeling text corpora, it can automatically cluster documents into mixtures of topics. Each topic is characterized by a distribution over words. In particular, the LDA model can automatically assign each document a probability distribution over topics and assign the topic distributions over the words. For example, different probabilities are assigned to the words for different topics. In our case, for the set of feedback documents to a given query, it may also contain plenty of topics.

The LDA model assumes the following generative process for each document $d$ in a document collection $D$:

1. Pick a multinomial distribution $\Phi_z$ for each topic $z$ from a Dirichlet distribution with hyperparameter $\beta$. $\beta$ is the parameter of the uniform Dirichlet prior on the per-topic word distribution.

2. For each document $d$, pick a multinomial distribution $\theta_d$ from a Dirichlet distribution with hyperparameter $\alpha$. $\alpha$ is the parameter of the uniform Dirichlet prior on the per-document topic distributions.

3. For each word $w$ in document $d$,

   (a) Choose a topic $z_n \sim Multinomial(\theta_d)$, where $n \in \{1, 2, \ldots, K\}$.

   (b) Choose the word $w$ from the multinomial distribution of $\varphi_{z_n}$.

Thus, the probability of generating the collection $D$ is given as following:

$$P(d_1, \ldots, d_{|D|}|\alpha, \beta) =$$
$$\iint \prod_{z=1}^{M} P(\varphi_z|\beta) \prod_{d=1}^{|D|} P(\theta_d|\alpha) (\prod_{i=1}^{N_d} \sum_{z_i=1}^{M} P(w_i|z, \varphi)) d\theta d\varphi$$

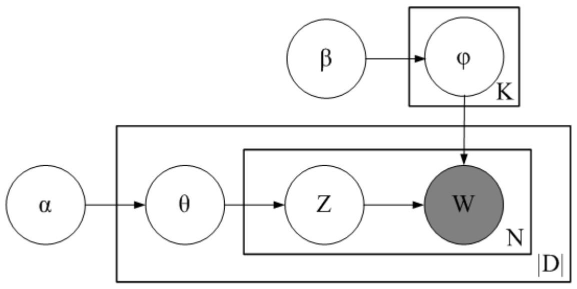


Figure 4.2: Plate notation for the LDA model

where $|D|$ is the number of documents in dataset $D$, $N_d$ is the number of words in document $d$, $K$ is the number of topics in the LDA model. Figure 4.2 depicts the plate notation for the LDA model, which can capture the dependencies among all the variables.

The LDA model is complicated and intractable to compute. Although the posterior distribution is intractable for exact inference, a wide variety of approximate inference algorithms can be considered for LDA, including Laplace approximate, variational approximation, Markov chain Monte Carlo [Wal04] and Gibbs sampling [GG84]. In this paper, we use LingPipe's [12] implementation of LDA, in which the LDA model is estimated using a simplified form of the Gibbs sampler [PNI+08]. The following probabilities can be generated from LDA: $P(z_k|d)$, $P(w|z_i)$ and $P(z_i)$ where $i \in \{1, 2...M\}$ and $M$ is the number of topics. In this article, we use $P(z_k|d)$ as the

[12]http://alias-i.com/lingpipe/

coordinates in the topic space. Documents are represented as vectors of length $M$, and evaluated by their topic representations for PRF in Section 4.4.2, 4.4.3, 4.4.4 and 4.4.5.

### 4.4.2 A Probabilistic Framework: TopPRF

As far as we know, little work has been done on how to integrate topical information into probabilistic PRF models naturally and effectively. So we implement our idea on the classic Rocchio's model and BM25 is utilized as the basic model. The effectiveness and flexibility of the Rocchio's model make it very suitable for extensions. So we decide to integrate the topic space information into it and propose a better framework. We name the new framework TopPRF and "Top" denotes the integration of topics.

The traditional Rocchio's model is based on the term vector space and it is proven to be effective. So we keep this term information and utilize it together with information from the topic space. Matches on the term level are accurate but rigid. On the topical level, matches can bring something different while they are actually processed on groups of terms instead of a particular one. So we consider the cooperation of the two spaces is promising. As we mentioned above, we plan to adjust the weights of candidate feedback terms according to the topical information we have. The TopPRF framework is shown as follows:

$$\vec{Q_1} = \alpha * \vec{Q_0} + \beta * \sum_{\vec{r} \in D_R} \frac{\vec{r} * TS(d_k)}{|R|} \tag{4.1}$$

where $\vec{Q_1}$ and $\vec{Q_0}$ represent the original and first iteration query vectors, $D_R$ is the set of pseudo relevance documents, $\vec{r}$ is the expansion term weight vector for each feedback document, $R$ is the set of feedback documents, $|R|$ is the number of feedback documents, $d_k$ is the $k$-th feedback document in $R$, $TS(d_k)$ is the score of feedback documents based on our proposed topic space methods.

In fact, we only add one factor into the Rocchio's model without importing new parameters. So we will not suffer the resource-consuming process of new parameter optimization and cost much time on sampling or grid search. Although we only propose three models as follows, the framework can be easily extended with different $TS(d_k)$. This will encourage researchers to discover more good models. In $\vec{r}$, we use *tf-idf* as in the traditional Rocchio's model. Particularly, we set $\alpha$ to 1 and train an optimal $\beta$ in the experiments. More details can be found in Section 4.5.3.

The complexity of each iteration of the Gibbs sampling for LDA is linear with the number of topics and the number of documents [WC06]. In our case, the time complexity is denoted as O($M$*$|R|$). For each query, we use at most 50 feedback documents and 20 topics in LDA, which is constant time O(1) to the size of the collection. Therefore, our proposed approaches do not bring higher computational complexity in theory. We will have more discussion about the time cost of our proposed approaches in the experiments.

### 4.4.3 TS-COS: Measuring Topic Similarity via Cosine Formula

When integrating LDA on the top $k$ documents $d_1, d_2, ...d_k$ in the retrieved list, we will have $M$ topics $z_1, z_2, ..., z_M$ and a $k * M$ Document-Topic matrix which contains the probability $p(z_i|d_n)$ , where $n$ is from 1 to k and $i$ is from 1 to M. For each document $d_n$, we consider it to be a probability vector of topics $p(z_1|d_n), p(z_2|d_n), ..., p(z_M|d_n)$ in the topic space, and the sum of all elements in this vector is 1. Because a document is the mixture of these M topics, the topic distribution can reveal the topic bias of this document. Moreover, if a relevant document of a particular query has a great bias towards some topics, e.g., $z_i$ and $z_j$, it is naturally to consider other documents with the same bias are more likely to be relevant.

We can use relevant documents as examples and measure the topic similarities between them and other documents. In our study, we first assume the top $s$ documents are relevant. Different from the top $k$ documents in traditional PRF, we attempt to ensure the $s$ documents to be really relevant, so we must choose a very small value for it. The $s$ documents are part of the feedback documents, and we use them to evaluate the reliability of the rest documents. We cannot guarantee all the $s$ documents are really relevant since the process is still pseudo. But according to our preliminary experiments on WT10G, smaller $s$ can lead to a higher ratio of relevant documents in the group. We consider this $s$ document group as the *trustable group.* Sometimes, the relevant documents for a particular query cover several topics. In order to maintain the balance of document relevance and topic diversity, we first choose three as a reasonable number for $s$ in our research, and then try different values to see how $s$ impacts the performance.

For the proposed method named TS-COS, we measure the similarity between the topic vectors of two documents via the cosine formula. Thus, the topic similarity of document $i$ and $j$ is as follows:

$$cos(d_i, d_j) = \frac{\sum\limits_{t=1}^{M} (p(z_t|d_i) * p(z_t|d_j))}{\sqrt{\sum\limits_{t=1}^{M} (p(z_t|d_i))^2} \times \sqrt{\sum\limits_{t=1}^{M} (p(z_t|d_j)^2}} \tag{4.2}$$

where $z_t$ is the t-th topic, M is the total topic number.

When we set $s = 3$, $TS(d_k)$ is calculated as follows:

$$TS(d_k) = \frac{\sum\limits_{i=1}^{3} cos(d_k, d_i)}{3} \tag{4.3}$$

where the scores of the top 3 documents are set to 1 since they are supposed to be relevant. $TS(d_k)$ is normalized to (0.5,1) to avoid a significant difference between two documents when $k > 3$.

Intuitively, we think all the feedback documents are somewhat relevant while they are the top-ranked ones obtained through the models like BM25. More or less, they have something related to one or more query terms. After all, the process of weighting feedback documents is still pseudo. We are not 100 percent sure about the relevance of the sample documents. So we set the floor of the final similarity scores to 0.5 to narrow down the differences between the best and the worst documents. The normalization is simple. We just divide the Cosine similarity by 2 and plus 0.5.

### 4.4.4 TS-EU: Measuring Topic Similarity via Euclidean Distance

In order to see how different similarity methods perform, we change the cosine similarity method to another popular one, Euclidean distance. For this method, we actually consider documents as the points in the topic space. The distance between two document points indicates their closeness, or similarity. The distance between document $i$ and $j$ in the topic space is as follows:

$$Euclidean(d_i, d_j) = \frac{\sqrt{\sum_{t=1}^{M} (p(z_t|d_i) - p(z_t|d_j))^2}}{M} \tag{4.4}$$

where $z_t$ is the t-th topic, M is the total topic number.

Equation 4.1 is still used to choose feedback terms. Unlike Equation 4.2, large Euclidean distance between two documents means they are not similar. So when $s = 3$, $TS(d_k)$ is calculated as follows:

$$TS(d_k) = 1 - \frac{\sum_{i=1}^{3} Euclidean(d_k, d_i)}{3 \times M} \tag{4.5}$$

We name this method TS-EU as EU represents Euclidean distance and $TS(d_k)$ is also normalized to [0.5, 1] as in Equation 4.3.

### 4.4.5 TS-Entropy: Measuring Document Purity via Topic Entropy

When we use the feedback documents for query expansion, the whole content of each document is supposed to be relevant, which is not true in most cases. Terms in the irrelevant part of a document can bring useless information and harm the performance of PRF. In general, we consider documents which contain few topics are more reliable for choosing feedback terms. In other words, terms in a "pure" feedback document are likely to be relevant than those from a document containing multiple topics. To measure the purity of a document, we import a new concept called *topic entropy*.

Entropy is an important concept in the information theory. It is a measure of the unpredictability of information content. Suppose a discrete random variable $X$ has possible values $x_1, ..., x_n$, the entropy of $X$ can be calculated as:

$$H(X) = \sum_i P(x_i)\ I(x_i) = -\sum_i P(x_i) \log_b P(x_i) \tag{4.6}$$

where b is the base of the logarithm used, and $I(x_i)$ is the information content of $x_i$ [Hay95].

If we consider a document $d$ as the variable $X$, its value depends on the topic it is about. So $x_i$ in Equation 4.6 is actually topic $z_t$ given $d$. The topic entropy of document $d$ is calculated as:

$$H(d) = \sum_{t=1}^{M} P(z_t|d)\ I(z_t|d) = -\sum_{t=1}^{M} P(z_t|d) \log_b P(z_t|d) \tag{4.7}$$

Large topic entropy means multiple topics have non-ignorable probability give $d$

and this document is not pure. Documents with small topic entropy should therefore be more reliable than others for PRF. Again we modify Equation 3.1 to apply this TS-Entropy method.

$TS(d_k)$ in Equation 4.1 is calculated as follows:

$$TS(d_k) = 1 - \frac{H(d)}{\log_b M} \tag{4.8}$$

where the value is normalized to [0, 1], b is set to 2 in this paper.

## 4.5 Experimental Settings

In this section, we describe the settings in our experiments. Experiments are conducted on five standard TREC data sets described as in Section 4.5.1. Section 4.5.2 presents three baselines models for comparison with our proposed approaches, including the classic BM25, two state-of-the-art RM3 and BM25-based Rocchio. In Section 4.5.3, we discuss how to set and optimize the parameters in the baselines, as well as how to train the parameters in our proposed approaches.

Table 4.2: Information about the test collections.

| Collection | Queries | # Docs |
|---|---|---|
| disk1&2 | 51-200 | 741,856 |
| disk4&5 | 301-450, 601-700 | 528,155 |
| WT2G | 401-450 | 247,491 |
| WT10G | 451-550 | 1,692,096 |
| GOV2 | 701-850 | 25,178,548 |

### 4.5.1 Collections

We evaluate our proposed methods on five public TREC [13] datasets with ad-hoc queries, including disk1&2, disk4&5, WT2G, WT10G and GOV2 which are different in size and genre [14]. The Disk1&2, Disk4&5 collections contain newswire articles from various sources, such as Association Press (AP), Wall Street Journal (WSJ), Financial Times (FT), etc., which are usually considered as high-quality text data with little noise. The WT2G collection is a general crawl of Web documents, which has 2 Gigabytes of uncompressed data. This collection was used in the TREC 8 Web track. The WT10G collection is a medium size crawl of Web documents, which was used in the TREC 9 and 10 Web tracks. It has 10 Gigabytes of uncompressed data. GOV2 is a very large crawl of the .gov domain, which has more than 25 million

[13]http://trec.nist.gov/

[14]In each chapter, the datasets and parameter settings can be slightly different because the experiments were done in different periods for different work, but the results are statistically meaningful because hundreds of queries were covered

documents with an uncompressed size of 423 Gigabytes. The TREC tasks and query numbers associated with each collection are presented in Table 5.1.

Queries for these datasets are provided by TREC in more than ten years, and the datasets are widely used for IR [ZHH11, CMSS14, CPL15]. We only use the title part of the queries to retrieve because users usually only input several keywords when searching in the real world. For the preprocessing of the collections, we use the Porter Stemmer [Por80] and general stopword remover [ACC$^+$00] with 418 stopwords removed.

### 4.5.2 Baselines in Comparison

In the experiments, we compare our proposed methods with the traditional probabilistic model BM25, and two state-of-the-art pseudo relevance feedback models, RM3 and Rocchio. These three baselines are described as follows.

BM25 is a famous traditional weighting model, which has been recognized good performance in IR. In BM25, the weight of a search term is assigned based on its within-document term frequency and query term frequency [RWJ$^+$94b]. The formula of BM25 has been introduced in Chapter 2. Rocchio's method has the following steps to incorporate (pseudo) relevance feedback information into the retrieval process [YH16], and its process is introduced in details in the previous Chapter.

RM3 is an interpolated version of relevance model [LC01b], which is a representative and state-of-the-art approach for re-estimating query language models for PRF [LZ09c]. Relevance language models do not explicitly model the relevant or pseudo-relevant document. Instead, they model a more generalized notion of relevance $R$. We also introduce the formula of RM1 in Chapter 2 as follows:

$$p(w|R) \propto \sum_{\theta_D} p(w|\theta_D)p(\theta_D)P(Q|\theta_D) \tag{4.9}$$

The relevance model $p(w|R)$ is often used to estimate the feedback language model $\theta_F$, and then interpolated with the original query model $\theta_Q$ in order to improve its estimation as follows:

$$\theta_{Q'} = (1-\alpha) * \theta_Q + \alpha * \theta_F \tag{4.10}$$

This interpolated version of relevance model is `RM3`. [LZ09c] systematically compare five state-of-the-art approaches for estimating query language models in ad-hoc retrieval, in which RM3 not only yields impressive retrieval performance in both precision and recall metric, but also performs steadily. In particular, we apply Dirichlet prior for smoothing document language models [ZL01a].

### 4.5.3 Parameter Settings and Optimization

Particularly, for the basic retrieval model, we use the BM25 model, and search optimal $b$ from 0.1 to 0.9 with the step 0.1. Also, we search $\beta$ from 0.1 to 0.9 with the step 0.1 for the Rocchio's model as well. Because we will evaluate the impact of topical information on the feedback document, we fix the feedback term number to 30. Feedback document numbers are set to 10, 20, 30 and 50 respectively. The number of topics is set to be 5, 10 and 20, which are reasonable when the feedback document number is not large. Mean Average Precision (MAP) is the metric for evaluations. The MAP metric reflects the overall accuracy and the detailed descriptions for it can be found in [VH00]. Language model with Dirichlet prior is used as the basic model for another baseline RM3. For the smoothing parameter $\mu$, we sweep over values from 500 to 2000 with an interval of 100. The interpolation parameter $\alpha$ for RM3 is set from 0.1 to 0.9 with the step 0.1. All the experimental results are evaluated through 2-fold cross-validation. The TREC queries are partitioned into two sets by the parity of TREC queries number on each dataset. Parameters trained on one set are applied to the other set and then vice versa for evaluation as in [YHL11b].

## 4.6 Experimental Results and Analysis

This section presents the experimental results and compares the proposed approaches with the baselines. In Section 4.6.1, we demonstrate that the classic BM25 is a reasonable basic model for PRF in our proposed approaches. For a fair comparison, BM25 is also adopted in Rocchio with the same settings. The performance of baseline models is shown in Section 4.6.2, and the performance of the proposed topic-space based approaches are discussed in Section 4.6.3. We study the impact of using LDA over the whole collection for PRF in Section 4.6.4. Further analyses of the experimental results are provided in Section 4.6.5.

### 4.6.1 Comparison of Basic Retrieval Models

As we mentioned in the previous section, the results of both models are obtained by 2-fold cross-validation with optimal parameters. It is therefore fair to compare them on these five collections. As shown in Table 4.3, BM25 slightly outperforms LM with Dirichlet prior on the Disk1&2 and WT2G collection. The results of these two models are almost the same over the Disk4&5, WT10G and GOV2 collections. This comparison indicates that the classic BM25 model is generally comparative to LM, and it is reasonable to use them as the basic models of the PRF baselines and our proposed methods.

Table 4.3: BM25 vs LM on the five TREC collections

| | disk1&2 | disk4&5 | WT2G | WT10G | GOV2 |
|---|---|---|---|---|---|
| BM25 | 0.2380 | 0.2494 | 0.3124 | 0.2055 | 0.3034 |
| LM | 0.2320 | 0.2510 | 0.2995 | 0.2063 | 0.3040 |

### 4.6.2 Performance of Baseline Models

Table 4.4: MAP obtained by the baselines, TS-COS, TS-EU and TS-Entropy. A "*" and a "+" symbol indicate a statistically significant improvement over the RM3 and the Rocchio baselines according to the Wilcoxon matched-pairs signed-ranks test at the 0.05 level. The percentage in the parentheses is the improvement over them. The best performance in each line is in bold.

| $\lvert D_f \rvert$ | BM25 | RM3 | Rocchio | TS-COS | TS-EU | TS-Entropy |
|---|---|---|---|---|---|---|
| | | | | disk1&2 | | |
| 10 | 0.2380 | 0.2665 | 0.2962 | **0.3019**$^{*+}$ | 0.3017$^{*+}$ | 0.3014$^{*+}$ |
| | | | | (13.28%, 1.92%) | (13.21%, 1.86%) | (13.10%, 1.73%) |
| 20 | 0.2380 | 0.2652 | **0.3095** | 0.3073$^{*}$ | 0.3056$^{*}$ | 0.3048$^{*}$ |
| | | | | (15.87%, -0.71%) | (15.23%, -1.26%) | (14.93%, -1.54%) |
| 30 | 0.2380 | 0.2632 | 0.2950 | **0.3075**$^{*+}$ | 0.3060$^{*+}$ | 0.3046$^{*+}$ |
| | | | | (16.83%, 4.24%) | (16.26%, 3.73%) | (15.73%, 3.15%) |
| 50 | 0.2380 | 0.2610 | 0.2953 | **0.3054**$^{*+}$ | 0.3035$^{*+}$ | 0.3022$^{*+}$ |
| | | | | (17.01%, 3.42%) | (16.28%, 2.78%) | (15.79%, 2.28%) |
| Average | 0.2380 | 0.2640 | 0.2990 | **0.3055** | 0.3042 | 0.3033 |
| | | | | (15.74%, 2.18%) | (15.24%, 1.74%) | (14.88%, 1.40%) |
| | | | | disk4&5 | | |
| 10 | 0.2494 | 0.2720 | 0.2876 | **0.3035**$^{*+}$ | 0.3020$^{*+}$ | 0.2979$^{*+}$ |
| | | | | (11.85%, 5.53%) | (11.03%, 5.01%) | (9.52%, 3.46%) |
| 20 | 0.2494 | 0.2709 | 0.2894 | **0.3028**$^{*+}$ | 0.2998$^{*+}$ | 0.2973$^{*+}$ |
| | | | | (11.78%, 4.63%) | (10.67%, 3.59%) | (9.75%, 2.66%) |
| 30 | 0.2494 | 0.2695 | 0.2801 | **0.2927**$^{*+}$ | 0.2898$^{*+}$ | 0.2868$^{*+}$ |
| | | | | (8.61%, 4.50%) | (7.53%, 3.46%) | (6.42%, 2.34%) |
| 50 | 0.2494 | 0.2576 | 0.2688 | **0.2824**$^{*+}$ | 0.2772$^{*+}$ | 0.2720$^{*+}$ |
| | | | | (9.63%, 5.06%) | (7.61%, 3.13%) | (5.59%, 1.18%) |
| Average | 0.2494 | 0.2675 | 0.2815 | **0.2954** | 0.2922 | 0.2885 |
| | | | | (10.41%, 4.93%) | (9.23%, 3.81%) | (7.85%, 2.44%) |
| | | | | WT2G | | |
| 10 | 0.3124 | 0.3244 | 0.3219 | **0.3261**$^{+}$ | 0.3214 | 0.3146 |
| | | | | (0.52%, 1.30%) | (-0.92%, -0.16%) | (-3.02%, -2.32%) |
| 20 | 0.3124 | 0.3255 | 0.3233 | 0.3338$^{*+}$ | **0.3379**$^{*+}$ | 0.3283$^{+}$ |
| | | | | (2.55%, 3.25%) | (3.81%, 4.52%) | (0.86%, 1.52%) |
| 30 | 0.3124 | **0.3222** | 0.2979 | 0.3198$^{+}$ | 0.3176$^{+}$ | 0.3076$^{+}$ |
| | | | | (-0.74%, 7.35%) | (-1.43%, 6.61%) | (-4.53%, 3.15%) |
| 50 | 0.3124 | 0.3234 | 0.3092 | **0.3131** | 0.3104 | 0.3049 |
| | | | | (-3.18%, 1.26%) | (-4.02%, 0.39%) | (-5.72%, -1.41%) |
| Average | 0.3124 | **0.3239** | 0.3131 | 0.3232 | 0.3218 | 0.3139 |
| | | | | (-0.21%, 3.23%) | (-0.63%, 2.79%) | (-3.10%, 0.25%) |
| | | | | WT10G | | |
| 10 | 0.2055 | 0.2164 | 0.2045 | 0.2172$^{+}$ | **0.2183**$^{+}$ | 0.2093$^{+}$ |
| | | | | (0.37%, 6.21%) | (0.88%, 6.75%) | (-3.88%, 2.29%) |
| 20 | 0.2055 | 0.2151 | **0.2193** | 0.2171 | 0.2102 | 0.2077 |
| | | | | (0.93%, -1.00%) | (-2.28%, -4.15%) | (-3.44%, -5.58%) |
| 30 | 0.2055 | **0.2123** | 0.1993 | 0.2078$^{+}$ | 0.2021 | 0.2007 |
| | | | | (-2.12%, 4.26%) | (-4.80%, 1.40%) | (-5.46%, 0.70%) |
| 50 | 0.2055 | **0.2098** | 0.2010 | 0.2008 | 0.1991 | 0.1926 |
| | | | | (-4.29%, -0.10%) | (-5.10%, -0.95%) | (-8.20%, -4.36%) |
| Average | 0.2055 | **0.2134** | 0.2060 | 0.2107 | 0.2074 | 0.2026 |
| | | | | (-1.25%, 2.28%) | (-2.80%, 0.68%) | (-5.07%, -1.70%) |
| | | | | GOV2 | | |
| 10 | 0.3034 | 0.3172 | 0.3343 | **0.3550**$^{*+}$ | 0.3532$^{*+}$ | 0.3498$^{*+}$ |
| | | | | (11.92%, 6.19%) | (11.35%, 5.65%) | (10.28%, 4.43%) |
| 20 | 0.3034 | 0.3167 | 0.3345 | **0.3578**$^{*+}$ | 0.3529$^{*+}$ | 0.3455$^{*+}$ |
| | | | | (12.98%, 6.97%) | (11.43%, 5.50%) | (9.09%, 3.18%) |
| 30 | 0.3034 | 0.3160 | 0.3354 | **0.3527**$^{*+}$ | 0.3449$^{*+}$ | 0.3463$^{*+}$ |
| | | | | (11.61%, 5.16%) | (9.15%, 2.83%) | (9.59%, 3.15%) |
| 50 | 0.3034 | 0.3138 | 0.3309 | **0.3491**$^{*+}$ | 0.3410$^{*+}$ | 0.3384$^{*+}$ |
| | | | | (11.25%, 5.50%) | (8.67%, 3.05%) | (7.84%, 2.22%) |
| Average | 0.3034 | 0.3160 | 0.3338 | **0.3537** | 0.3480 | 0.3450 |
| | | | | (11.94%, 5.95%) | (10.15%, 4.26%) | (9.20%, 3.25%) |

All the experimental results are shown in Table 4.4 and Figure 4.3 . $|D_f|$ is the number of feedback documents for PRF. Rocchio in Table 4.4 actually denotes Rocchio's model with BM25 as the first-pass retrieval model, and RM3 denotes LM+RM3. As we can see from Table 4.4, the Rocchio's model generally outperforms BM25 in most cases. On these five collections, the Rocchio's model achieves its best performance on 4 of the five collections and the second best performance on the GOV2 collection when $|D_f|$ is 20. When more feedback documents are chosen, the performance drops dramatically and even worse than BM25 on WT2G and WT10G. This indicates that documents are more and more unreliable when their ranks are lower.

RM3, which is a state-of-the-art model PRF for language modeling, generally outperforms the Rocchio's model on the WT2G and WT10G collections, but not very significantly. This indicates that the Rocchio's model is still a very strong baseline for IR research work. Compared to the Rocchio's model, the results of RM3 are more stable when the number of feedback documents changes.

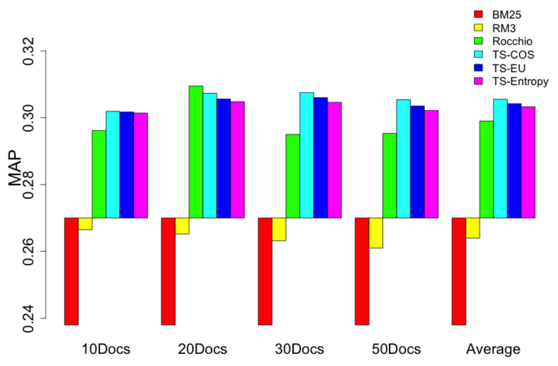
### 4.6.3 Performance of Topic-Space Based Models

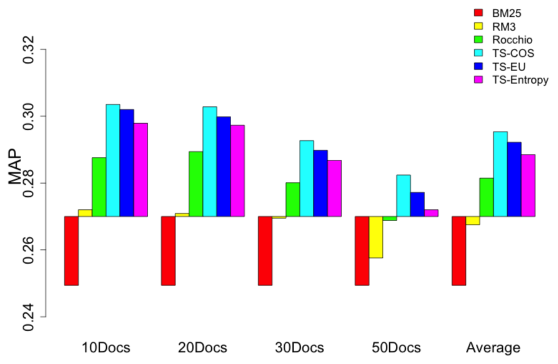
Although our proposed methods are all based on the Rocchio's model, we can see their performance is quite different. In most cases, TS-COS achieves the best results. It is only surpassed by RM3 on the WT10G collection while its base model, Rocchio, does not work well. Even under that condition, the average performance of TS-COS is just slightly weaker than RM3's. Also, its average performance outperforms that of the Rocchio's model on all collections significantly. These experimental results justify the effectiveness of the TS-COS method and the application of topic similarity.

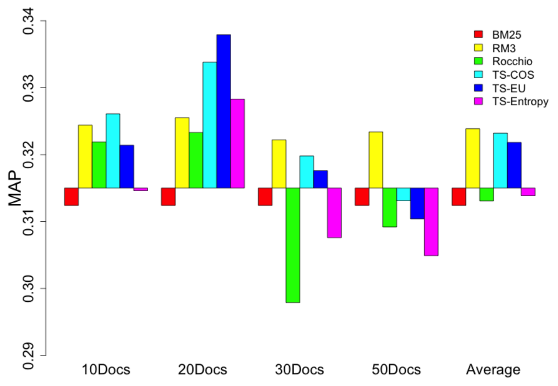
In Figure 4.3, we have a more clear view on the general performance of each method[15]. The performance of TS-EU changes with TS-COS. It is a little worse than TS-COS except in three cases. Although they are based on the same idea, their performance is different. The similarity model we choose can affect the overall results significantly. Consequently, it is possible to have better performance if we investigate more similarity models. On average, TS-EU also outperforms Rocchio's on all the collections.

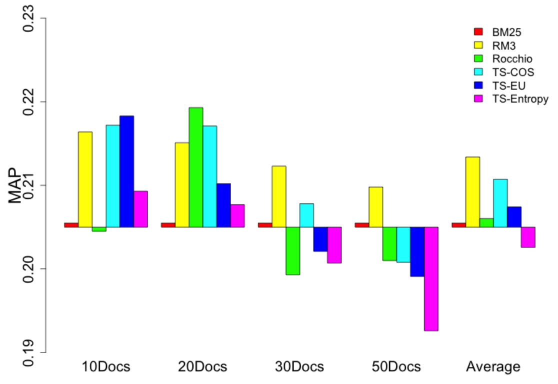
Different from the other two methods, the TS-Entropy method is not so outstanding. But its results do verify our assumptions. It is better than the Rocchio's model on 4 out of 5 collections while its results are usually worse than TS-COS and TS-EU. With the increase of collection size, the performance gap between TS-Entropy and the other two methods become larger and larger. In a large collection, the ratio of irrelevant documents is larger as well. So the purity of documents will not be helpful to identify irrelevant documents. Nevertheless, the weights of pure relevant docu-

[15]In order to show the performance clearly, we choose a value around the average performance of all the methods as the base.


BM25
RM3
Rocchio
TS-COS
TS-EU
TS-Entropy
MAP
0.24
0.26
0.28
0.30
0.32
10Docs
20Docs
30Docs
50Docs
Average


BM25
RM3
Rocchio
TS-COS
TS-EU
TS-Entropy
MAP
0.24
0.26
0.28
0.30
0.32
10Docs
20Docs
30Docs
50Docs
Average


BM25
RM3
Rocchio
TS-COS
TS-EU
TS-Entropy
MAP
0.29
0.30
0.31
0.32
0.33
0.34
10Docs
20Docs
30Docs
50Docs
Average



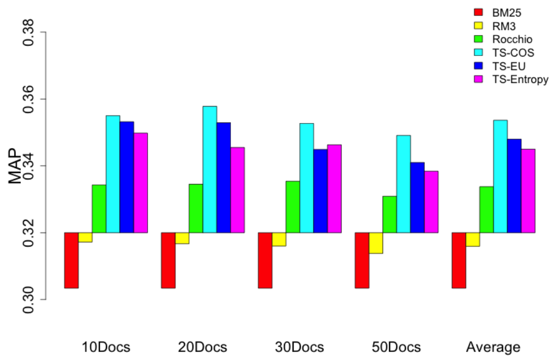


Figure 4.3: Performance of BM25, RM3, Rocchio, TS-COS, TS-EU, TS-Entropy methods on the 5 TREC collections

ments are still enhanced by the method, and mostly this factor makes improvements. Its performance highly depends on its base, the Rocchio's model. On WT10G, the average performance of the Rocchio's model is marginally better than BM25, and the MAP of BM25 is only 0.2055. In that case, the average performance of the TS-Entropy method is 1.70% worse than that of the Rocchio's model. On the contrary, the Rocchio's model obtain the best performance on GOV2 with MAP 0.3338, and the TS-Entropy method obtains the largest improvement on average (3.25%) over it. So the experimental results justify that those relevant documents which are "pure" in topics can help improve the overall performance of PRF. Generally, TS-Entropy is useful when Rocchio performs well, and this feature can be used in other text mining applications to evaluate the diversity of a document.

The performance of all the three methods has similar trends in different cases while they are implemented under the same framework TopPRF. In Figure 4.3, we can see that the bars of these three methods grow up or bend down almost synchronously when the conditions change. For instance, if TS-COS obtain the best result when $|D_f|$ is 30, the other will also get their best performance in this situation. TS-COS is mostly the best one. TS-EU is a little worse than TS-COS, and TS-Entropy is always the worst among three. On disk1&2, disk4&5 and GOV2, it is obvious that all the three methods are much better than RM3. Their base model Rocchio contributes some while its performance is better than RM3, too. On WT10G, however, the bad performance of Rocchio also pulls down the three methods while TS-COS can be better than RM3 sometimes. Although the proposed methods are generally better than their base Rocchio, they are affected by the performance of Rocchio significantly because they use the same feedback documents. While we can obtain solid improvements by costing little in adjusting the weights of these documents, the experimental results are still encouraging.

### 4.6.4 Impact of Using LDA with the Whole Collection for PRF: A Case Study

One reason why we choose only part of the feedback documents for topic modeling instead of the whole collection is the issue of time complexity. As we have mentioned in Section 4.4.2, when we only choose a fixed small set of documents for topic modeling and a fixed small number of topics, the time complexity of each iteration for LDA is O(1). However, the time complexity of using the whole collection will be linear with the number of topics and the number of documents $O(M * N)$, where $M$ is the number of topics and $N$ is the size of the collection. Here we have conducted a case study of using the whole collection on a relatively small dataset Disk 1&2. We spent 15 hours on building the topics offline on our server with Intel(R) Xeon(R)

CPU E5410 @ 2.33GHz, 32G RAM. Our proposed approaches do not need this time for building topic space, since it costs constant time for building it online. The time spent in our experiments is comparable to Rocchio. According to our experiments on the GOV2 collection for 150 queries, Rocchio takes 2,056 seconds [16] and TS-COS takes 2,391 seconds with 20 topics. TS-COS takes about 10% time more than the Rocchio's model. In general, the time complexity for our proposed methods is quite reasonable when only the top documents are used in building LDA.

Regarding the the retrieval performance, we conduct experiments on using LDA with the whole collection for PRF and compare with our strategy of using top documents. Experiments are conducted on Disk 1&2 with TREC queries 51-100. The MAP values of these two scenarios with different feedback document numbers and topic numbers are shown in Table 4.5. We can observe that the results are very similar. The best result for each feedback document number is bolded, and using LDA on top documents has better performance than using LDA on whole collection. The reason can be that using the whole collection will consider a boarder area of topics, which may not be related to the given query. On the other hand, building LDA on top documents will make the topic space less sparse and therefore the differences between relevant and irrelevant documents become more obvious. In other words,

[16] The experiments are conducted on our Intel(R) Core(TM) i7-2600 CPU, 8G RAM workstations. The time cost may change with different environments.

more focused topics will be generated, which are more potentially to be relevant to the query. Similar trends can be observed on the other datasets.

Table 4.5: MAP comparison between building topic space from topic documents and whole collection. Experiments are conducted on Disk 1&2 with TREC queries 51-100. All topic spaces are built via LDA.

| | LDA on the Top Documents | | | LDA on the Whole Collection | |
|---|---|---|---|---|---|
| topic / $\|D_f\|$ | 5 | 10 | 20 | 100 | 200 |
| 10 | 0.2747 | 0.2744 | **0.2756** | 0.2734 | 0.2691 |
| 20 | 0.2757 | 0.2767 | **0.2769** | 0.2762 | 0.2700 |
| 30 | 0.274 | **0.2753** | 0.275 | 0.2751 | 0.2680 |
| 50 | 0.2734 | **0.2738** | 0.2732 | 0.2718 | 0.2690 |

### 4.6.5 Analyses

The improvements made by our methods come from the combination of pseudo relevance feedback and topic modeling. The term-based pseudo relevance feedback model, BM25-based Rocchio (Section 4.5.2), performs significantly better than the basic weighting model, BM25. For example, Rocchio has 10.2% improvement (from 0.3034 to 0.3343 in Table 4.4) over BM25 on data set GOV2. If we look at topic modeling approaches in the past, Yi and James [YA08] explored several different types of topic models for retrieval purpose, and their experimental results indicated that none of the topic model approaches could outperform RM on any data set. Moreover, from Table 4.4, we can see that BM25-based Rocchio generally performs better than RM3.

Therefore, none of the pure topic modeling approaches can significantly outperform Rocchio. Our proposed methods, incorporating topic space into feedback, can bring further contribution for boosting performance in most of the cases, compared to either pseudo relevance feedback only approaches or topic modeling only approaches. For example, TS-COS significantly improves BM25-based Rocchio by 6.19% (from 0.3343 to 0.3550 in Table 4.4) on GOV2. Further, we have also studied the impact of using LDA with the whole collection for PRF. It is observed that using only top documents is more efficient and effective than using the whole collection.

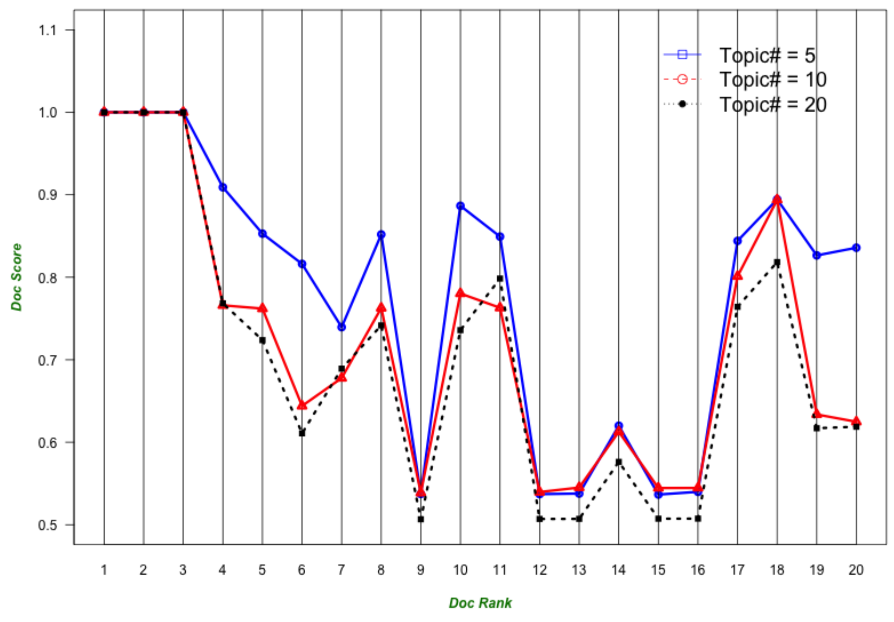
Besides the challenge of identifying topics, we consider another latent problem is the loss of topical information. Using a few topics can neglect useful information in other ones, even when they are relevant to the query topic. For example, terms with higher probabilities in the selected topic(s) are considered to be more important for PRF. However, they can also appear frequently in other topics and they are not so informative. In that case, terms which have even probabilities in many topics should be less important. Without the information of full topics, we will miss this kind of features. Choosing particular topics is a kind of dimension reduction. When topics are not stable, it is better to keep all the topic information. Therefore, we do not consider other dimension reduction methods like PCA or NMF in the scope of this paper.

## 4.7 Further Experiments and Discussions

In this section, we will conduct an in-depth discussion and analysis based on our experimental results. At first, we will show two case studies on the same queries used in Section 4.3.1 and demonstrate how the ranks of feedback documents change with the topic number. As a result, our proposed model can achieve stable performance based on the relative ranks of these documents. Next, we will compare TS-COS with the state-of-the-art Topk_LDA method. The experimental results indicate that the performance of our proposed method is much more stable and at least as good as the latter. Finally, we will show how the size of the trust group affects the performance.

### 4.7.1 Discussions of Two Case Studies

As we show in Section 4.3.1, the number of topics has an impact on the term distributions in the "most related" topic extensively. In this section, we will demonstrate how the topic number influences our proposed methods. Particularly, we make TS-COS as the representative of our three proposed methods. Both TS-EU and TS-Entropy perform similarly to TS-COS.



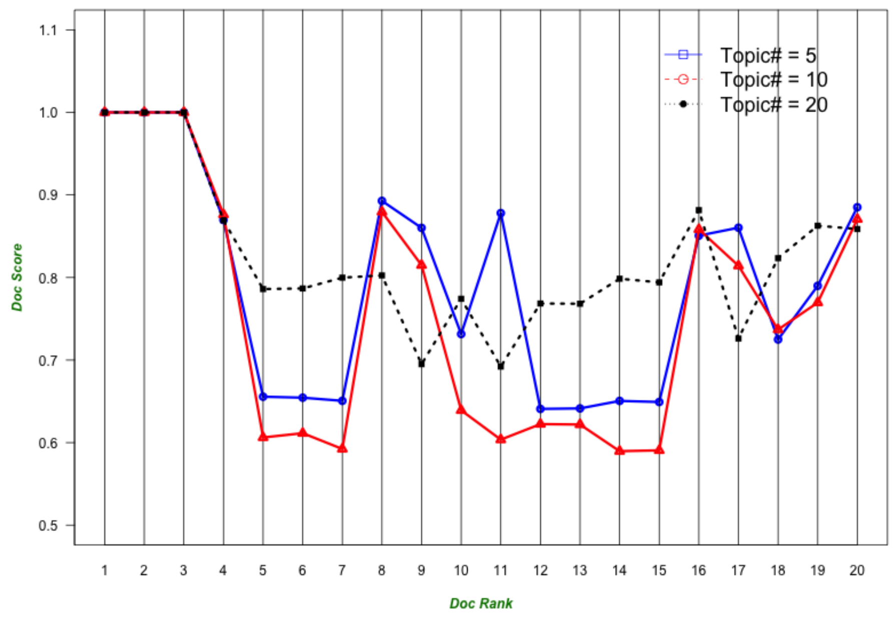


Figure 4.4: Cosine similarity scores of top 20 documents for query 802 and 804

To make a fair comparison, we use the same parameter settings as in Section 4.3.1. Query 802 and 804 are still used as examples. To demonstrate the results more clearly, we plot the scores of the top 20 documents graphically in Figure 4.4. The score for each document with different topic numbers can be found in Appendix A. As we can see from Figure 4.4, the score curves do not fall monotonously. Some documents obtain higher scores than those which are above them. That indicates documents with lower ranks in the retrieved list can be similar to the trustable group on the topical level and assigned more weights.

When the topic number increases, the curves become more smooth. While there are more topics or we can say topics are more fine-grained, the topic distributions of two documents have more trivial differences. Consequently, the maximum similarity score of the feedback documents (except the trustable group) will be smaller when the topic number increases, and the range of the scores will be narrower while we normalize them to be above 0.5. In summary, the topic number can be used to adjust the differences among feedback documents.

For query 802, the score rank of each document is almost unchanged for different topic numbers. But it is not that perfect for query 804. For instance, the score of document 10 is higher than document 11 when the topic number is 5, but is lower than the latter when the topic number is 10 or 30. With the changes of topic number, the score trends of top $k$ documents are generally stable, especially when compared with the term ranks in Table 4.1.

The difference between the curves for query 802 and 804 can be caused by the quality of the trustable group. The P@3 of BM25 for query 802 is 1.0, which means all the documents in the trustable group are relevant. Our proposed TS-COS method benefits from the high quality of the top 3 documents. Query 802 "Volcano eruptions global temperature" is also very clear. So the related information in the top 3 documents should be very close. If a document is similar to the first document, it will be similar to the other two. In other words, documents which are relevant can consistently gain high scores when the topic number changes because they will always get high similarity scores from each of the top 3 documents. The differences among documents are very clear. Compared with the performance of BM25 (AP 0.3241) and Rocchio (AP 0.3939), TS-COS obtains 0.4079, 0.4080 and 0.4084 respectively for topic number 5, 10 and 30.

Meanwhile, the P@3 result of BM25 for query 804 is only 0.3333, and only the third document in the trustable group is relevant. Relevant documents have very similar information which makes it easy to identify them from the irrelevant ones. On the contrary, irrelevant sample documents are not very helpful to identify other irrelevant documents, since their contents can be totally different. In this case, the relative ranks of feedback documents can be unstable when the topic number changes due to the unpredictable matching. When the topic number increases, most documents will not have overlaps on the topics of the irrelevant samples and obtain close scores. That can explain why the curve becomes smoother when the topic number is 20. An interesting phenomenon is that the performance of TS-COS is not bad than the Rocchios' model in this case. The average precision (AP) performance of Rocchio is 0.5632, and TS-COS gets 0.5762, 0.5761 and 0.5759 respectively for topic number 5, 10 and 20. A possible reason is that when the trustable group is not good, the score differences of these documents are not huge. So terms are compared mainly on their term features in the feedback documents. That ensures the performance will not be much worse than the Rocchio's model. Also, only the relevant sample can be helpful to identify feedback documents.

In summary, our proposed methods can provide very good and stable results when the samples are relevant. If most of the samples in the trustable groups are not relevant, our proposed method can prevent the performance from dropping too much by taking the term features into account. At the same time, the score range of these documents will be narrower due to the diversity of irrelevant information. In this case, the impact of topical information is reduced when evaluating the weight of feedback terms.

### 4.7.2 Comparisons with Topk_LDA

To support our argument that the performance is more robust by integrating topic space into PRF under our proposed probabilistic framework, we compare one of our methods TS-COS with the best method Topk_LDA in [YHL11b] and see how the performance changes according to topic numbers on five standard TREC collections. Topk_LDA is a state-of-the-art approach in integrating topical information on PRF, which chooses a set of top topics with weights higher than a given threshold and select terms based on their probabilities given these topics.

To make fair comparisons, we set feedback term numbers in {10, 20, 30, 40, 50} as in [YHL11b]. This is different from the setting in Section 4.6, where the feedback term number is fixed to be 30. The rest of the settings are the same as described in Section 4.5.3. The comparison results with Topk_LDA are shown in Table 4.6, 4.7, 4.8, 4.9 and 4.10. Since more feedback term numbers are screened, these results are slightly better than those in Table 4.4, but the trends are the same. Also, [YHL11b] uses a different query set from our experiments, and here we implement our approach TS-COS to the query set in [YHL11b] in Table 4.7 for a fair comparison.

First of all, the performance of Topk_LDA shows big difference when topic number changes. For example in Table 4.6 on Disk1&2, Topk_LDA loses about 8% performance when the topic number is changed from 5 to 20 for the feedback document

Table 4.6: The performance change of Topk_LDA and TS-COS on Disk1&2 when topic number is 5, 10 and 20. The percentage in the parentheses is the designated MAP over the MAP for topic number 5. "*" indicates a statistically significant improvement over Topk_LDA according to the Wilcoxon matched-pairs signed-ranks test at the 0.05 level.

| | Topk_LDA | | |
|---|---|---|---|
| topic / $\lvert D_f \rvert$ | 5 | 10 | 20 |
| 10 | 0.2897 | 0.2871 (-0.90%) | 0.2648 (-8.60%) |
| 20 | 0.2967 | 0.2893 (-2.49%) | 0.2813 (-5.20%) |
| 30 | 0.2963 | 0.2897 (-2.22%) | 0.2818 (-4.89%) |
| 50 | 0.2941 | 0.2912 (-0.99%) | 0.2876 (-2.21%) |
| | TS-COS | | |
| topic / $\lvert D_f \rvert$ | 5 | 10 | 20 |
| 10 | 0.3056* | 0.3055* (-0.03%) | 0.3060* (-0.20%) |
| 20 | 0.3108* | 0.3112* (0.13%) | 0.3108* (0.00%) |
| 30 | 0.3107* | 0.3106* (-0.03%) | 0.3110* (-0.10%) |
| 50 | 0.3087* | 0.3087* (0.00%) | 0.3085* (-0.64%) |

Table 4.7: The performance change of Topk_LDA and TS-COS on Disk4&5 when topic number is 5, 10 and 20. To compare with Topk_LDA, we only use the same queries 301-450. So the performance is quite different from what we show in 4.4. The percentage in the parentheses is the designated MAP over the MAP for topic number 5. "*" indicates a statistically significant improvement over Topk_LDA according to the Wilcoxon matched-pairs signed-ranks test at the 0.05 level.

| | Topk_LDA | | |
|---|---|---|---|
| topic / $\lvert D_f \rvert$ | 5 | 10 | 20 |
| 10 | 0.2581 | 0.2522 (-2.29%) | 0.2396 (-7.17%) |
| 20 | 0.2628 | 0.2579 (-1.86%) | 0.2523 (-4.00%) |
| 30 | 0.2631 | 0.2555 (-2.89%) | 0.2490 (-5.36%) |
| 50 | 0.2569 | 0.2527 (-1.63%) | 0.2506 (-2.45%) |
| | TS-COS | | |
| topic / $\lvert D_f \rvert$ | 5 | 10 | 20 |
| 10 | 0.2635 | 0.2622* (-0.49%) | 0.2616* (-0.72%) |
| 20 | 0.2625 | 0.2644 (0.72%) | 0.2634* (0.34%) |
| 30 | 0.2553 | 0.2540 (-0.51%) | 0.2538*(-0.66%) |
| 50 | 0.2498 | 0.2506 (0.32%) | 0.2488 (-0.4%) |

number 10. In Table 4.9 on WT10G, Topk_LDA gained about 4% when changing topic number from 5 to 20 for feedback document number 30. However, our proposed

Table 4.8: The performance change of Topk_LDA and TS-COS on WT2G when topic number is 5, 10 and 20. The percentage in the parentheses is the designated MAP over the MAP for topic number 5. "*" indicates a statistically significant improvement over Topk_LDA according to the Wilcoxon matched-pairs signed-ranks test at the 0.05 level.

| | Topk_LDA | | |
|---|---|---|---|
| topic / $\lvert D_f \rvert$ | 5 | 10 | 20 |
| 10 | 0.3171 | 0.3031 (-4.41%) | 0.3091 (-2.52%) |
| 20 | 0.3161 | 0.3082 (-2.50%) | 0.3039 (-3.86%) |
| 30 | 0.3170 | 0.3121 (-1.55%) | 0.3130 (-1.26%) |
| 50 | 0.3174 | 0.3147 (-0.85%) | 0.3129 (-1.42%) |
| | TS-COS | | |
| topic / $\lvert D_f \rvert$ | 5 | 10 | 20 |
| 10 | 0.3204 | 0.3202 (0.06%) | 0.3208 (0.18%) |
| 20 | 0.3384 | 0.3384* (0.00%) | 0.3385* (0.03%) |
| 30 | 0.3197 | 0.3207 (0.31%) | 0.3192 (-0.16%) |
| 50 | 0.3112 | 0.3112 (0.00%) | 0.3117 (0.16%) |

Table 4.9: The performance change of Topk_LDA and TS-COS on WT10G when topic number is 5, 10 and 20. The percentage in the parentheses is the designated MAP over the MAP for topic number 5.

| | Topk_LDA | | |
|---|---|---|---|
| topic / $\lvert D_f \rvert$ | 5 | 10 | 20 |
| 10 | 0.2310 | 0.2283 (-1.17%) | 0.2297 (-0.56%) |
| 20 | 0.2290 | 0.2289 (-0.04%) | 0.2333 (1.87%) |
| 30 | 0.2220 | 0.2285 (2.93%) | 0.2325 (4.73%) |
| 50 | 0.2267 | 0.2273 (0.26%) | 0.2312 (1.99%) |
| | TS-COS | | |
| topic / $\lvert D_f \rvert$ | 5 | 10 | 20 |
| 10 | 0.2157 | 0.2224 (3.11%) | 0.2212 (2.55%) |
| 20 | 0.2204 | 0.2148 (-2.54%) | 0.2152 (-2.41%) |
| 30 | 0.2004 | 0.2062 (2.89%) | 0.2059 (2.74)% |
| 50 | 0.2020 | 0.1988 (-1.58%) | 0.1967 (-2.62%) |

method TS-COS is much more robust. The results are not sensitive respect to the topic numbers.

Table 4.10: The performance change of Topk_LDA and TS-COS on GOV2 when topic number is 5, 10 and 20. The percentage in the parentheses is the designated MAP over the MAP for topic number 5. "*" indicates a statistically significant improvement over Topk_LDA according to the Wilcoxon matched-pairs signed-ranks test at the 0.05 level.

| | Topk_LDA | | |
|---|---|---|---|
| topic / $\|D_f\|$ | 5 | 10 | 20 |
| 10 | 0.3445 | 0.3327 (-3.43%) | 0.3352 (-2.69%) |
| 20 | 0.3446 | 0.3333 (-3.28%) | 0.3357 (-2.58%) |
| 30 | 0.3443 | 0.3335 (-3.14%) | 0.3322 (-3.51%) |
| 50 | 0.3488 | 0.3473 (-0.43%) | 0.3412 (-2.18%) |
| | TS-COS | | |
| topic / $\|D_f\|$ | 5 | 10 | 20 |
| 10 | 0.3589* | 0.3580* (-0.25%) | 0.3576* (-0.36%) |
| 20 | 0.3605* | 0.3587* (-0.50%) | 0.3588* (-0.49%) |
| 30 | 0.3559 | 0.3546 *(-0.37%) | 0.3541* (-0.51%) |
| 50 | 0.3506 | 0.3496 (-0.29%) | 0.3486 (-0.57%) |

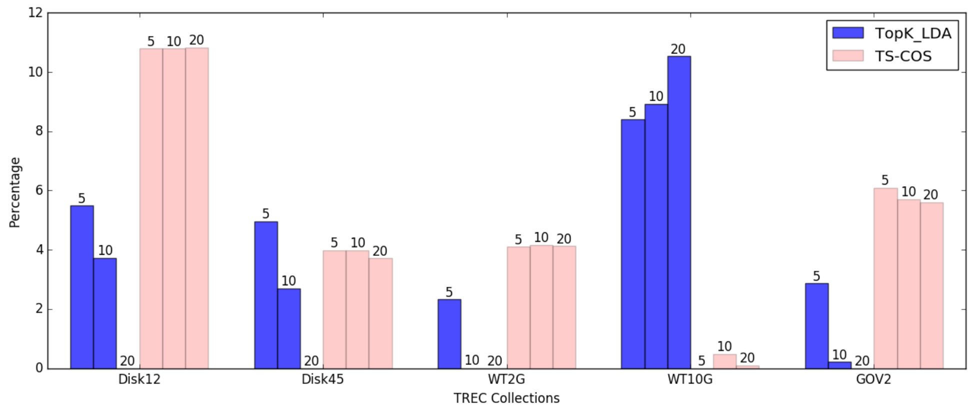


Figure 4.5: Comparisons between Topk_LDA and TS-COS with different number of topics. Results are percentages based on the lowest value on each collection.

Figure 4.5 shows the performance of Topk_LDA and TS-COS on the five TREC collections. All results are averaged based on the number of feedback documents, and converted to percentages based on the lowest value on each collection. We can observe that Topk_LDA and TS-COS behave very differently when the topic number changes from 5 to 20. Topk_LDA's performance is highly sensitive to the change of topic numbers. On the other hand, TS-COS usually has very similar performance with different topic numbers. These results show that the usage of topic space can reduce the sensitivity to the topic number when integrating full topic-document information for PRF. This is due to the reason that the feedback terms' weights are adjusted based on the scores of corresponding feedback documents. Top-ranked documents are more likely to be relevant to the query, and the terms appearing in the top documents are more likely to be good feedback terms. Our proposed approaches keep the highly-weighted documents and evaluate the reliability of other feedback documents that can potentially provide more relevant terms according to the proposed topic-based approaches. The feedback term list keeps the good feedback terms in the top documents, and is expanded by more relevant terms based on the term weight and the reliability of the feedback document. Therefore, the feedback term list is stable no matter how the topics are changed with different topic numbers. On the other hand, approaches only relying on selected topics, such as Topk_LDA, changes the feedback term set according to the topics. And the good feedback terms can be neglected if the topics are not well selected. That is the reason why Topk_LDA could be significantly affected by the change of generated topics.

Also, the TS-COS method generally outperforms Topk_LDA significantly on four out of five collections, Disk12, Disk45, WT2G and GOV2. Although we have used Rocchio in our approaches, which has better performance than RM3 [17] on these two collections, these experimental results justify that our idea can make solid improvements over strong baselines. On WT10G, Topk_LDA has better performance than TS-COS. The reason could be that the feedback documents from the first pass retrieval on WT10G are more irrelevant to the query, since the MAP on WT10G is low. Thus, there is more noise in the generated topics. Topk_LDA method has removed the noisy topics in the feedback process. TS-COS can capture all topical information and could affect by the noisy topics. However, our proposed TS-COS has more advantages for further improving the basic models with high accuracies. When the basic model does not perform well, TS-COS can also improve the retrieval performance. But the improvement is not as high as the case that the basic model has good performance.

To summarize, the integration of topic space in PRF makes the performance of our proposed methods more robust than methods like the state-of-the-art Topk_LDA. When the fluctuation caused by the "fuzzy topics" is relieved, the application of topic space can enhance the performance of the classic Rocchio's model. When using topics

[17]Actually, the performance of RM3 in [YHL11b] is better than that in this paper because Ye et al. tried more parameter values, e.g. term numbers. The performance of TS-COS in 4.6, 4.7, 4.8, 4.9 and 4.10 is better than that in Table 4.4 due to the same reason.

to represent documents and mining latent relations among documents from them, the overall performance can be more stable than methods selecting particular topics. Our proposed methods will not filter topical information even when it is identified as "irrelevant". Information hidden in all topics can be useful when measuring the similarity between two documents. If two documents have quite similar distributions over all topics, we will know they are similar in semantics and do not have to identify which topics are really relevant. It is better to identify how relevant a document is rather than a topic while the latter is not stable with different topic numbers. At the same time, significant improvements over the strong baseline BM25-based Rocchio's model also shows that the integration of topical information can benefit the term-based PRF by importing information on a different grade. Thus, the integration of topic space brings both robustness and significant improvements over strong baseline models. We can conclude that topic space is a very beneficial complementary for traditional term-based matching. In the future, it is promising to integrate topic space into other topic modeling applications.

### 4.7.3 Trustable Group Size $s$

In this section, we will discuss how the size of the trustable group $s$ affects the performance of the topic-similarity based methods. Because TS-EU performs similarly

Table 4.11: Impact of P@n of the basic model BM25 on the TS-COS method which shows the ratio of relevant documents in the top $n$ list, the best performance under each condition is in bold

| $\|D_f\|$ | 1 | 2 | 3 | 5 | 10 |
|---|---|---|---|---|---|
| | | | disk1&2 | | |
| 10 | 0.3014 | 0.3009 | **0.3019** | 0.3016 | NA |
| 20 | 0.3066 | 0.3069 | **0.3073** | 0.3062 | 0.3056 |
| 30 | 0.3071 | **0.3076** | 0.3075 | 0.3071 | 0.3070 |
| 50 | 0.3049 | 0.3051 | **0.3054** | 0.3046 | 0.3043 |
| P@n | 0.5733 | 0.5267 | 0.5244 | 0.5187 | 0.5053 |
| | | | disk4&5 | | |
| 10 | 0.3021 | 0.3029 | **0.3035** | 0.3007 | NA |
| 20 | 0.2997 | 0.3014 | **0.3028** | 0.3002 | 0.3017 |
| 30 | **0.2935** | 0.2929 | 0.2927 | 0.2926 | 0.2926 |
| 50 | 0.2822 | 0.2824 | 0.2824 | 0.2831 | **0.2836** |
| P@n | 0.5582 | 0.5321 | 0.5261 | 0.4980 | 0.4345 |
| | | | WT2G | | |
| 10 | 0.3181 | 0.3127 | **0.3261** | 0.3183 | NA |
| 20 | 0.3371 | 0.3343 | 0.3338 | **0.3377** | 0.3327 |
| 30 | 0.3196 | **0.3226** | 0.3198 | 0.3157 | 0.3199 |
| 50 | 0.3123 | **0.3136** | 0.3131 | 0.3121 | 0.3133 |
| P@n | 0.5800 | 0.5800 | 0.5400 | 0.5040 | 0.4840 |
| | | | WT10G | | |
| 10 | 0.2164 | 0.2157 | **0.2172** | 0.2143 | NA |
| 20 | **0.2176** | 0.2084 | 0.2171 | 0.2107 | 0.2076 |
| 30 | **0.2088** | 0.2085 | 0.2078 | 0.2044 | 0.2005 |
| 50 | 0.2008 | **0.2015** | 0.2008 | 0.2013 | 0.2003 |
| P@n | 0.4900 | 0.4400 | 0.4200 | 0.3840 | 0.3280 |
| | | | GOV2 | | |
| 10 | **0.3556** | 0.3527 | 0.3550 | 0.3531 | NA |
| 20 | 0.3580 | **0.3594** | 0.3578 | 0.3544 | 0.3493 |
| 30 | 0.3540 | 0.3537 | 0.3527 | **0.3543** | 0.3477 |
| 50 | 0.3503 | 0.3525 | **0.3537** | 0.3516 | 0.3461 |
| P@n | 0.6970 | 0.6465 | 0.6431 | 0.6182 | 0.5818 |

to TS-COS, we only focus on TS-COS.

To investigate the impact, we also demonstrate the ratio of the relevant documents in the trustable group which is P@n of the basic model BM25. We set the size of the group $s$ to 1, 2, 3, 5 and 10. In total, we have four different $|D_f|$ 10, 20, 30 and 50 for all the 5 collections. We consider the combination of a particular $|D_f|$ and a particular collection as a certain condition, and therefore we have 20 conditions to compare the performance of TS-COS with different sizes of trustable groups. All the results are shown in Table 4.11.

From the table, we can see P@n decreases when $n$ is larger. This evidence shows that the quality of the trustable group does fall if we use more documents. P@10 of BM25 is less than 0.5 on 3 out of 5 collections, and only a little higher than that on the disk1&2 dataset. Accordingly, only under one condition does TS-COS obtain the best result with a 10-document trustable group (on disk4&5 when $|D_f|$ is 50). So it is better to choose a small $s$ for similarity calculation.

Furthermore, although P@1 is usually much high than other P@n results, TS-COS does not benefit much from it. The 1-document group performs the best in 4 out of 20 conditions, but none of these four results are significantly better than that obtained when $s$ is 3. Generally, the performance of TS-COS using different $s$ is close on disk1&2, disk4&5 and WT2G. On WT10G, we obtain significant improvements when $s$ is 1 and three over others. This indicates that when the overall quality of the trustable group is not good, i.e., P@n is comparatively low, the performance of TS-COS is very sensitive to the values of $s$. When $s$ is 10, the performance drops significantly than the best performance. Meanwhile, 3 is a good choice under eight conditions. This justifies the assumption that the relevant topics are not covered by the first feedback document in many cases.

Generally, it is better to choose a small $s$, especially when the results obtained through the first pass retrieval is not good. To take the diversity of the relevant topics into account, 1 is not the best choice in most cases. According to the experimental results in this Section, it is safe to set $s$ to be 3 for different datasets.

# Chapter 5

# High Performance Query Expansion Using Adaptive Co-training

we propose to incorporate an adaptive co-training method for selecting feedback documents which is a substantial extension of [HHW$^+$06a]. In [HHW$^+$06a], an initial study of applying co-training for QE was proposed over a small TREC HARD track dataset with 23 queries for passage retrieval. The basic idea of this method is to initialize a co-training process by taking the top-ranked and bottom-ranked passages in initial retrieval as positive and negative examples, respectively. The features representing the passages are randomly split into two sets, on which two different classifiers are learned respectively to label the remaining documents. Despite the limited but encouraging improvement on the TREC HARD track dataset, the approach [HHW$^+$06a] for selecting feedback documents using co-training suffers from the following issues. First, the random feature division may lead to the situation when the discriminative power of the features is completely unbalanced in the two sets. In this case, one of the classifiers may not be properly learned due to the low-quality features it has. Second, the previous approach is unable to cope with queries for which the top-ranked documents in the initial retrieval results are rather poor for training classifiers. The co-training process in [HHW$^+$06a] will stop only when a particular number of iterations have been done. Thus, new feedback documents will be added even if the trained classifiers are of poor quality. As a result, the new feedback documents are not reliable and could bring negative effects into the QE process. In this case, we should stop the co-training process to avoid unreliable feedback documents. Hence, The AUC measure [WF05] is introduced for monitoring the quality of the learned classifiers[18]. AUC is calculated by using approximations of integrals. It adds up the area of all trapezoids (green rectangles and yellow triangles under the curve ) as shown in Figure 1 [19].

The iterative co-training process ends when AUC is lower than a given threshold. This adaptive criterion is different from that in the previous work [HHW$^+$06a, XA08], whose halting condition of iterations is only determined by a predefined value. Details about how to set this threshold will be discussed in Section 5.4. Compared to this preliminary work, this work includes additional learning models, extensive experiments on more datasets, more systematic result analyses, more comprehensive discussion and more conclusive findings.

[18]The Area Under a ROC Curve (AUC), calculated by the Mann-Whitney statistics, is a standard measurement for the soundness of a classification [WF05].The value of AUC ranges between 0 and 1. A high AUC value indicates a success in the classification and a low value indicates the contrary

[19] `http://en.wikipedia.org/wiki/File:Integral_approximations.svg`

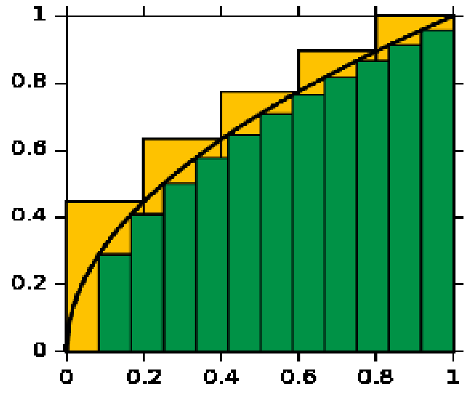


Figure 5.1: AUC calculation

## 5.1 Adaptive Co-Training For QE

The standard co-training method assumes that an instance comes with two complementary sets of features in nature. For example, the features that describe a Web page can be the words on the page and the links that point to that page [BM98]. However, for text retrieval, it is not obvious how to derive two complementary sets of features to represent the feedback documents, while the link information is usually not available.

There are quite a few options that we can apply to represent the feedback documents. In summary, these options can be grouped into two main categories, namely the term-based features, and the high-level features.

The term-based document representation characterizes a feedback document by its composing terms. A typical example is to represent a feedback document by a vector of the expansion term weights in the candidate feedback documents, where the weight of an expansion term is estimated by its normalized document generation probability [Roc71a, Sal71]. In contrast to the term-based document representation, other related studies use relatively high-level features, such as the proximity of the expansion term and the original query terms, the co-occurrences of the expansion term and the original query terms in the collection, and so on [CNGR08, HO09a]. The high-level features are usually considered having a higher descriptive power of the feedback documents than the term-based features, as they are found to be important factors affecting QE's effectiveness [CNGR08, HO09a]. However, there is also difficulty to apply the high-level features for co-training in practice: it is expensive to compute the features such as the proximity of the expansion term and the original query terms, for each retrieved document. In particular, since such features are query-dependent, which means their values change from query to query, they have to be computed during retrieval time. This is infeasible for large-scale collections. Therefore, in this work, we rather follow the term-based document representation.

For $D$, the set of retrieved documents for a given query, we rank unique terms in $D$ by a descending order of their Kullback-Leibler Divergence (KLD) weights. KLD is a popular choice of expansion term weighting, which has been shown to be effective in many state-of-the-art QE methods [Ama03a, YHHL09]. For example, a KLD-based QE mechanism provides the best statMAP over the standard feedback sets in the TREC 2009 Relevance Feedback track [YHHL09]. KLD measures how a term's distribution in the feedback documents diverges from its distribution in the whole collection. The higher KLD is, the more informative the term is. For a unique term in $D$, the KLD weight is given by:

$$KLD(t) = P(t|D) \log_2 \frac{P(t|D)}{P(t|C)} \tag{5.1}$$

where $P(t|D) = \frac{c(t,D)}{c(D)}$ is the generation probability of term $t$ from $D$. $c(t, D)$ is the frequency of $t$ in $D$, and $c(D)$ is the count of words in $D$. $P(t|C) = \frac{c(t,C)}{c(C)}$ is the collection model. $c(t, C)$ is the frequency of $t$ in collection $C$, and $c(C)$ is the count of words in the whole collection $C$.

The *maxN* terms with the highest KLD weights in $D$ are taken as the features to represent the documents. In each document $d$ in $D$, the weight of a feature term $t$ is again given by the Kullback-Leibler divergence of the term's distribution in $d$ from its distribution in the whole collection. *maxN* is a parameter, which is obtained by tuning over a set of training topics. Once the feedback documents are represented by the feature terms which belong to the set $F$ in Figure 5.3, they are labeled by the AdapCOT method that is described in the next section.

### 5.1.1 Adaptive Co-training

In this section, we devise an adaptive mechanism to apply co-training over a small set of training data which contains a few positive and negative documents in this case. Positive documents are those of good quality, and the negative ones are on the opposite side. In the proposed iterative co-training process, the halting criterion automatically adapts to the quality of the learned classifiers. It monitors the quality of the learned classifiers at the end of each iteration using the AUC measure. The co-training process stops when the quality of the learned classifiers is below a given threshold, which will be discussed later in this work. The basic idea of our proposed method is to represent the documents by important terms in the retrieved set, which are highly weighted. A small number of top-ranked and bottom-ranked documents from the initial retrieval are used as positive and negative examples, respectively. The top-ranked documents are assumed to be of high quality, and provide good coverage on the query topic in their content. In contrast, the bottom-ranked documents are used as negative examples in which the content is assumed to be off-topic. A learning procedure is designed to label the retrieved documents so that the documents selected for QE are meant to be highly similar to the positive set, while being dissimilar to the negative set.

The proposed method learns two different classifiers from two non-overlapped sets of features which represent the documents. More specifically, it allows the two classifiers, C1 and C2, to label the unlabeled instances, i.e. the rest of the retrieved documents, for each other through an iterative process until one of the halting criteria is met. The positively labeled documents are then used for feedback in QE.

**Input**

L - a set of co.init.p top-ranked and co.init.n

bottom-ranked feedback documents

U - a set of unlabeled feedback documents

C1, C2 - two classifiers

F - a set of feature terms

Control parameters: co.init.p, co.init.n, co.p,

co.n, co.K, co.AUC

**Output**

L - an expanded set of labeled feedback documents

Figure 5.2: Symbol notations for the adaptive co-training algorithm

Figure 5.3 gives a general description of the proposed AdapCOT method, and Figure 5.2 explains the meaning of symbols used in Figure 5.3. The proposed method is based on the initial retrieval results returned by the search engine. The *co.p* top-ranked and the *co.n* bottom-ranked documents are used as the labeled positive and negative examples, respectively. The rest of the retrieved documents remain unlabeled. As explained in the previous section, *maxN* unique terms with the highest KLD weights are used as features to represent the retrieved documents. To facilitate the co-training process, these *maxN* terms are split into two different sets. Each term acts as one feature of the documents in our AdapCOT method. Instead of a random split suggested in [CKP04], we rank these *maxN* terms in decreasing order of their KLD weights, and group them into an odd-ranked set and an even-ranked set, respectively. One of the advantages of our split method over the random split is that our method balances the quality of terms in the two feature sets[20], and avoids the case that one of the classifiers is not properly learned due to an extremely unbalanced split.

[20]In query expansion, terms with high weights are usually considered as being of high quality, and are therefore added to the query.

**Method**

```
1. Split F into two feature sets F1 and F2

2. Do the following for co.K iterations:

  (1) Learn classifier C1 from L using F1 to rep-
  resent the documents in L
  (2) If the resulting AUC value is lower than
  co.AUC, break;
  (3) Use C1 to classify examples in U based on F1
  (4) Select co.p/co.n most confidently predicted
  positive/negative instances from U
  (5) Add these co.p + co.n instances with their
  corresponding labels into L
  (6) Remove these co.p + co.n examples from U
  (7) Learn classifier C2 from L using F2 to rep-
  resent the documents in L
  (8) If the resulting AUC value is lower than
  co.AUC, break
  (9) Use C2 to classify examples in U based on F2
  (10) Repeat steps (4) - (6)
```

Figure 5.3: The adaptive co-training algorithm (AdapCOT).

Once the feature terms are split into two sets, the two classifiers are learned on the labeled documents one by one, and label the remaining unlabeled documents for each other. Such a co-training process ends when one of the halting criteria is met. In our proposed method, we introduce two halting criteria as follows. First, the co-training process ends after co.K iterations. Second, the co-training process ends if AUC, a measure used for evaluating the successfulness of the classification over the labeled examples, is below a given threshold co.AUC. The second criterion is a key step in the proposed method. It ensures that the learned classifiers are good enough to classify the unlabeled documents. It is particularly useful when the initial retrieval is of a poor quality, and the "bad" feedback documents won't be labeled as positive since the co-training process does not proceed further when the classifiers are not properly learned. Note that the AUC value is computed using the documents labeled by the learned classifiers, and the actual relevance assessment information is not used. Weka[21], a powerful data mining tool, is used to calculate the value of AUC for the implementation of our AdapCOT.

Furthermore, in order to guarantee the quality of the labeled positive documents for relevance feedback, we introduce the following restrictions [HO09b]:

- Only the 50 top-ranked documents in the initial retrieval can be added to the

[21]www.cs.waikato.ac.nz/ml/weka

labeled positive document set. This is based on the empirical observation that QE is unlikely to benefit from a feedback document that is roughly ranked lower than 50.

- When two labeled positive documents receive an identical confidence value from the classifier, the one ranked higher is preferred. This is also based on the empirical observation that the higher ranked documents are likely to be better feedback documents than the lower ranked ones.

In addition, our proposed AdapCOT method automatically labels the most confidently predicted positive documents for relevance feedback. It is inexpensive to implement since the co-training process does not involve any relevance assessment information by human effort.

## 5.2 Classifiers

In our studies, we choose to apply three different learning methods to facilitate the proposed adaptive co-training method. These methods are among the most popular machine learning methods applied in IR, including the low-cost Naive Bayes and Logistic Regression, and the relatively sophisticated support vector machines. Since our work in this work is a significant extension of [HHW$^+$06a], we only test the combinations of two different classifiers as in [HHW$^+$06a]. Thus, we have three combinations as shown in Table 5.2.

Naive Bayesian (NB) is a simple statistical learning algorithm based on Bayes' theorem with strong independence assumptions. Despite their naive design and simple independence assumptions, NB classifiers have worked quite well in many complex real-world situations [WF05]. In this work, we apply the popular Gaussian kernel density function as follows:

$$P(x|C_i) = \frac{1}{\sigma_i\sqrt{\pi}} \exp\big(\frac{-(x-\mu_i)^2}{2\sigma_i}\big) \tag{5.2}$$

where the probability $P(x|C_i)$ of observing an instance $x$ in class $C_i$ is approximated by a Gaussian density function with a standard deviation $\sigma_i$ and mean $\mu_i$.

Logistic Regression (LR) is a generalized linear model for binomial regression [WF05]. The basic idea of LR is to apply a *logit* function to scale membership values, which may not be proper probabilities, to values between 0 and 1:

$$f(z) = \frac{e^z}{e^z+1} = \frac{1}{1+e^{-z}} \tag{5.3}$$

where a membership value $z$ is mapped to a value between 0 and 1.

Support vector machines (SVM) [Joa99] are widely used in text classification in recent years. Its underlying principle is structure risk minimization. Its objective is to determine a classifier or regression function which minimizes the empirical risk (i.e., the training set error) and the confidence interval (which corresponds to the generalization or test set error). Given a set of training data, an SVM determines a hyperplane in the space of possible inputs. This hyperplane will attempt to separate the positives from the negatives by maximizing the distance between the nearest positive examples and and negative examples. There are several ways to train SVMs. One particularly simple and fast method is the Sequential Minimal Optimization [Pla99] which is adopted in our study. In addition, we apply the non-linear homogeneous polynomial kernel function at degree $m$ as follows:

$$k(x_i, x_j) = (x_i \cdot x_j)^m \tag{5.4}$$

where $x_i$ and $x_j$ are real vectors in a $p$-dimensional space, and $p$ is the number of features. The exponential parameter $m$ is set to 1 as a default value in our study. In the following sections, the proposed co-training method is evaluated through extensive experiments.

## 5.3 Experimental Setup

Table 5.1: Information about the test collections.

| Coll. | TREC Task | Topics | # Docs |
|---|---|---|---|
| disk1&2 | 1-3, Ad-hoc | 51-200 | 741,856 |
| disk4&5 | 2004, Robust | 301-450, 601-700 | 528,155 |
| WT10G | 9, 10 Web | 451-550 | 1,692,096 |
| .GOV2 | 2004-2006 Terabyte Ad-hoc | 701-850 | 25,178,548 |
| ClueWeb09 | 2009 Relevance Feedback | rf.01-rf.50 | 49,375,681 |

We evaluate our proposed AdapCOT method on most of the existing TREC datasets with ad-hoc topics, including the disk1&2, disk4&5, WT10G, .GOV2, and ClueWeb B collections. Basic statistics about the test collections and topics are given in Table 5.1. Details about the collections can be found in previous chapters.

Each topic contains three topic fields, namely title, description and narrative. We only use the title topic field that contains very few keywords related to the topic. The title-only queries are usually as short as a realistic snapshot of real user queries in practice [XA08, ZL01c]. On each collection, we evaluate our proposed model by a 10-fold cross-validation. The test topics associated to each collection are randomly split into ten equal subsets. In each fold, we use 9 subsets of test topics for training, and use the remaining subset for testing. The overall retrieval performance is averaged over all 10 test subsets of topics. We use the TREC official evaluation measures in our experiments, namely the statMAP at 1000 on ClueWeb09 [APY06], and the Mean Average Precision (MAP) at 1000 on the other four collections [Voo05]. For each query, the top *co.init.p* and the bottom *co.init.n* ranked documents are used as the initial labeled examples for the co-training. Only the top 1000 retrieved documents are involved in the co-training process[22]. All statistical tests are based on Wilcoxon matched-pairs signed-rank test at the 0.05 level.

In our experiments, we apply Okapi's BM25 for document ranking. BM25 is a classical probabilistic model based on approximation to the assumed two-Poisson term frequency distribution [RWHB+95]. The equation of BM25 can be found in Section 2.1.1. In this work, parameter $b$ is obtained by Simulated Annealing [KGV83] over the training topics on the collection used [RZT04].

We apply the KLD weighting for query expansion [YHHL09]. The applied QE algorithm considers the top-ranked documents as the feedback set $D_f$. An expansion weight $w(t)$ is assigned to each unique term in $D_f$. $w(t)$ is the mean of KLD weights computed over each feedback document by Equation 5.1. Particularly, $D$ in Equation

[22]That is, a document ranked at 1000 is considered the bottom-ranked.

5.1 refers to the selected feedback documents in $D_f$.

The $|expT|$ terms with the highest mean KLD weights over $D_f$ are added to the query, where $expT$ is the set of expansion terms. In our experiments, we set $|expT|$, the number of expansion terms to be added to the query, to 20, as it is shown to be effective in some previous studies [BC06, CT09, KSCC11].

We mainly report on the results obtained by setting *co.init.p*, the initial number of top-ranked documents used as positive examples, to 2,3,4,5,8,10,15 and 20. Impact of varying *co.init.p* on AdapCOT's retrieval performance is also discussed later. In addition, *co.init.n*, the number of bottom-ranked documents used as negative examples, is set to the twice of *co.init.p*, i.e. $co.init.n = co.init.p \cdot 2$. This is based on the findings in previous study [HHW$^+$06a] that it would better have more initial negative examples than the positive ones. Moreover, in the experiment conducted by Blum & Mitchell [BM98] for classifying university Web pages, *co.p* and *co.n* are set to 1 and 3, respectively. That is, in each iteration, each classifier is allowed to add 1 new positive and 3 new negative examples to $L$. In this work, in order to reduce the number of parameters for tuning, we follow the setting of *co.p* and *co.n* recommended in [BM98]. Moreover, $co.K$, the number of iterations in each co-training process, is set to 3 according to the recommendation in [HHW$^+$06a]. Thus, at most 3 positive and 9 negative examples are added to $L$. If $co.K$ is 0, AdapCOT acts in the same way as traditional QE methods do. Other control parameters, namely the number of terms used for document representation (*maxN*), and the threshold $co.AUC$, are obtained over the training data in the cross-validation process. The related evaluation results are presented in the next section.

## 5.4 Experimental Results

We evaluate our proposed AdapCOT for selecting feedback documents against different baselines, namely the initial retrieval using BM25, query expansion (QE) using top-ranked documents, and the baseline co-training method (baseCOT) without the adaptive quality control proposed for AdapCOT.

### 5.4.1 Comparison with Initial Retrieval

In the first step of our experiments, Table 5.2 compares AdapCOT's retrieval performance with the an initial retrieval baseline using BM25. In this step, the setting of *co.init.p* is default to 3. As we can see from this table, AdapCOT markedly outperforms the BM25 baseline in all cases. This is not of a great surprise since AdapCOT applies QE, which usually improves the initial retrieval performance. Moreover, although the use of different classifiers leads to very similar retrieval performance, in the rest of the work, only the results obtained by using LR and SVM for co-training are presented because this combination gets four best results over all the five collections.

Table 5.2: MAP/statMAP obtained by the BM25 baseline and AdapCOT. A star indicates a statistically significant improvement over the baseline.

| Coll. | BM25 | NB+LR | NB+SVM | LR+SVM |
|---|---|---|---|---|
| disk1&2 | 0.2307 | 0.2795*, 21.15% | 0.2797*, 21.24% | 0.2797*, 21.24% |
| disk4&5 | 0.2499 | 0.2910*, 16.45% | 0.2916*, 16.69% | 0.2916*, 16.69% |
| WT10G | 0.2090 | 0.2338*, 11.87% | 0.2333*, 11.63% | 0.2338*, 11.87% |
| .GOV2 | 0.3041 | 0.3245*, 6.71% | 0.3243*, 6.64% | 0.3235*, 6.38% |
| ClueWeb09 | 0.2052 | 0.2288*, 11.50% | 0.2285*, 11.35% | 0.2330*, 13.55% |

In addition, we examine the accuracy of the co-training method in selecting feedback documents. Regarding the "goodness" of a feedback document, we consider the following three different categories: 1. By using the document alone for relevance feedback, a "good" feedback document leads to at least 5% improvement over AP, the average precision of the query in initial retrieval; 2. A "bad" feedback document leads to a decrease in AP by at least 5%; 3. Other feedback documents are considered "neutral".

Table 5.3 presents the percentage of the feedback documents selected by AdapCOT in the above three categories. Overall, a moderate accuracy of the AdapCOT method is observed. A major proportion of the feedback documents picked up by AdapCOT, about 2 out of 3, fall into either "good" or "neutral" categories. The remaining "bad" feedback documents may hurt the retrieval performance. However, the impact of "bad" feedback documents could be neutralized since they are to some degree outnumbered by the "good" ones. In the rest of this section, we show that the moderate accuracy of AdapCOT indeed leads to improved retrieval performance over strong baselines.

| Coll. | Good | Neutral | Bad |
|---|---|---|---|
| disk1&2 | 59.02% | 2.05% | 40.78% |
| disk4&5 | 56.16% | 5.48% | 38.36% |
| WT10G | 57.14% | 14.28% | 28.57% |
| .GOV2 | 54.17% | 16.67% | 29.17% |
| ClueWeb09 | 61.78% | 19.11% | 19.11% |
| Total | 60.27% | 6.54% | 33.33% |

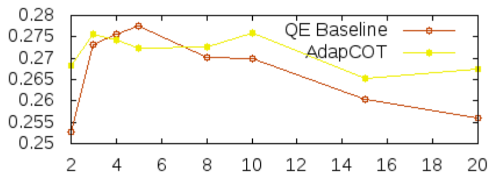
Table 5.3: The proportion of good/bad feedback documents picked up by the AdapCOT method.

### 5.4.2 Comparison with QE

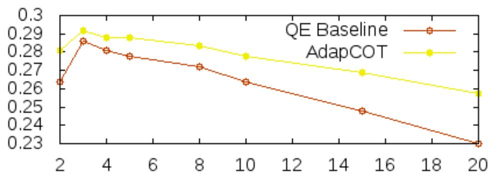
Table 5.4: MAP/statMAP obtained by the QE baseline and AdapCOT. A star indicates a statistically significant improvement.

| $\lvert D_f \rvert$ | QE | AdapCOT | QE | AdapCOT | QE | AdapCOT |
|---|---|---|---|---|---|---|
| | disk1&2 | | disk4&5 | | WT10G | |
| 2 | 0.2528 | 0.2684*, 6.17% | 0.2639 | 0.2808*, 6.40% | 0.2206 | 0.2332*, 5.95% |
| 3 | 0.2732 | 0.2797, 2.57% | 0.2857 | 0.2916, 2.06% | 0.2320 | 0.2338, 0.776% |
| 4 | 0.2757 | 0.2742, -0.182% | 0.2812 | 0.2877, 2.31% | 0.2209 | 0.2247, 1.72% |
| 5 | 0.2775 | 0.2725, -1.80% | 0.2775 | 0.2877, 3.68% | 0.2109 | 0.2216*, 5.07% |
| 8 | 0.2703 | 0.2727, 0.888% | 0.2723 | 0.2833, 4.04% | 0.1824 | 0.2007*, 10.03% |
| 10 | 0.2699 | 0.2759, 2.22% | 0.2636 | 0.2775*, 5.27% | 0.1782 | 0.1974*, 11.11% |
| 15 | 0.2603 | 0.2652, 1.88% | 0.2476 | 0.2689*, 8.60% | 0.1612 | 0.1852*, 14.89% |
| 20 | 0.2561 | 0.2675, 4.45% | 0.2301 | 0.2571*, 11.73% | 0.1412 | 0.1768*, 25.21% |
| | .GOV2 | | ClueWeb09 | | Average | |
| 2 | 0.2689 | 0.3038*, 12.98% | 0.1787 | 0.2303*, 28.88% | 0.2370 | 0.2633, 11.10% |
| 3 | 0.3003 | 0.3235*, 7.73% | 0.2047 | 0.2330*, 13.82% | 0.2591 | 0.2679, 3.28% |
| 4 | 0.2958 | 0.3218*, 8.79% | 0.2083 | 0.2116, 1.58% | 0.2564 | 0.2630, 2.57% |
| 5 | 0.2847 | 0.3165*, 11.17% | 0.1898 | 0.2000*, 5.37% | 0.2481 | 0.2597, 4.68% |
| 8 | 0.2644 | 0.3042*, 15.05% | 0.1863 | 0.2013*, 8.05% | 0.2351 | 0.2524, 7.36% |
| 10 | 0.2538 | 0.2946*, 16.07% | 0.1747 | 0.1948*, 6.58% | 0.2280 | 0.2480, 8.77% |
| 15 | 0.2375 | 0.2850*, 20.00% | 0.1777 | 0.1728, -2.76% | 0.2169 | 0.2354, 8.53% |
| 20 | 0.2325 | 0.2798*, 20.34% | 0.1680 | 0.1542, -8.21% | 0.2056 | 0.2271, 10.46% |

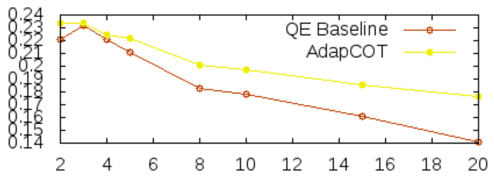
Next, we compare our proposed AdapCOT method with the QE baseline. QE is based on the assumption that the top-ranked documents are relevant, from which the most important terms are useful for retrieving more relevant documents. The size of the feedback document set has considerable impact on QE's effectiveness. In contrast, the AdapCOT method follows a slightly different assumption. It assumes the positivity of a small set of top-ranked documents, and the negativity of another small set of bottom-ranked documents. It selectively adds other retrieved documents to the positive document set through an iterative process until a halting criterion is met. Therefore, we compare the retrieval performance of AdapCOT with QE using the same number of top-ranked documents ($|D_f|$) for relevance feedback, and we vary $|D_f|$ from 2 to 20 to see how it influences the results of our AdapCOT and normal QE. The setting of *co.init.p*, the number of initial positive examples for co-training, is set to $|D_f|$. Meanwhile, we use the same number of expansion terms for both AdapCOT and QE, which is set to 20 as mentioned in Section 5.3. The related results are given in Table 5.4. In addition, Figure 5.4 plots the retrieval performance of the QE baseline and AdapCOT against different settings of $|D_f| = co.init.p$. An encouraging conclusion drawn from Table 5.4 and Figure 5.4 is that AdapCOT in general outperforms the QE baseline with different $|D_f|$ settings. AdapCOT performs particularly well on the large-scale .GOV2 and ClueWeb09 collections, where it achieves statistically significant improvement over the QE baseline even if $|D_f|$ is optimal for QE. AdapCOT also appears to be more robust than the QE baseline. In most cases, AdapCOT improves the QE baseline with different $|D_f|$ settings, even if the QE baseline has very poor retrieval performance. Besides, in Table 5.2 and Table 5.4, optimized BM25 outperforms the QE baseline and the AdapCOT in some cases, but generally QE and AdapCOT obtain better results when $|D_f|$ is optimal. Moreover, the best result of the QE baseline on .GOV2 (MAP 0.3003) is not so good as that of BM25 (MAP 0.3041), whereas AdapCOT (MAP 0.3243) still surpasses BM25 significantly when $|D_f|$ is set to 3. This indicates the robustness of our AdapCOT method.



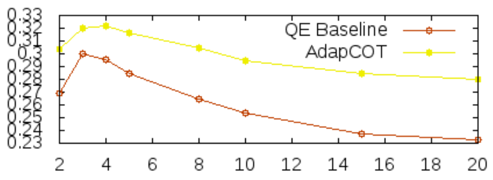
(a) Disk1&2



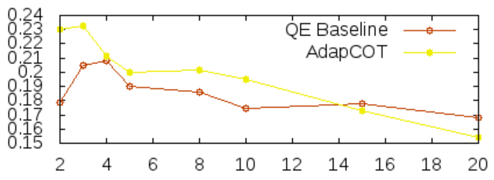
(b) Disk4&5



(c) wt10g



(d) gov2



(e) cluewebB

Figure 5.4: The MAP/statMAP obtained by AdapCOT and by the QE baseline (Y-axis) with different settings of $|D_f| = co.init.p$ (X-axis) on collection disk1&2, disk4&5, WT10G, .GOV2 and ClueWeb09.

Table 5.5: The percentages of "good" feedback documents selected by the QE baseline and AdapCOT

| $\|D_f\|$ | QE | AdapCOT | QE | AdapCOT | QE | AdapCOT |
|---|---|---|---|---|---|---|
| | disk1&2 | | disk4&5 | | WT10G | |
| 2 | 62.68 | 55.67 | 47.60 | 37.74 | 46.50 | 41.67 |
| 3 | 62.00 | 59.02 | 45.38 | 56.16 | 40.33 | 47.14 |
| 4 | 60.34 | 52.90 | 43.68 | 37.74 | 37.50 | 43.33 |
| 5 | 59.60 | 49.90 | 43.13 | 40.54 | 35.60 | 50.00 |
| 8 | 57.33 | 53.56 | 40.44 | 36.58 | 32.88 | 48.65 |
| 10 | 57.00 | 53.12 | 38.59 | 32.14 | 31.60 | 33.15 |
| 15 | 54.94 | 50.65 | 35.80 | 23.81 | 28.13 | 32.44 |
| 20 | 53.33 | 47.53 | 32.89 | 28.57 | 27.05 | 29.63 |
| | .GOV2 | | ClueWeb09 | | Average | |
| 2 | 20.92 | 46.43 | 33.68 | 55.67 | 42.28 | 47.44 |
| 3 | 30.15 | 54.17 | 31.97 | 61.78 | 41.97 | 60.27 |
| 4 | 39.38 | 51.43 | 35.20 | 65.54 | 43.22 | 55.65 |
| 5 | 49.23 | 53.33 | 33.88 | 63.97 | 44.29 | 51.55 |
| 8 | 37.46 | 41.38 | 35.98 | 62.61 | 40.82 | 48.56 |
| 10 | 44.15 | 50.00 | 36.73 | 58.50 | 41.61 | 45.38 |
| 15 | 42.10 | 58.62 | 35.92 | 55.57 | 39.38 | 44.22 |
| 20 | 40.31 | 68.42 | 35.61 | 57.90 | 37.84 | 46.41 |

Furthermore, we examine whether using AdapCOT would give a higher percentage of good feedback documents than just taking the top-ranked documents in Table 5.5. Although this is indeed the case on some of the collections used, it is surprising that the retrieval performance of both the QE baseline and AdapCOT is not necessarily correlated with the percentage of good feedback documents when $|D_f|$ becomes large. By further analysis on the experiments, we discover that the effectiveness of QE depends heavily on some key documents, which have the most important contribution to the expanded query. Therefore, if these key documents are included in the feedback set, the retrieval performance is usually good as long as the good feedback documents are not outnumbered by the bad ones. In this case, AdapCOT is particularly useful when the key documents are not highly ranked, which would not be picked up the QE baseline. A typical example is query 195 on disk1&2, for which the initial average precision (AP) obtained by BM25 is 0.0273. For this query, the document with id AP891017-0231 has the most contribution to the expanded query. However, as this key document is only ranked 35th in the initial retrieval, it is not used for feedback by the QE baseline. In contrast, AdapCOT manages to automatically identify this key document and add it to the feedback document set. It is then of no surprise that the QE baseline is unable to improve over the initial retrieval, while AdapCOT provides an AP of 0.1638 with a 500% improvement.

Figure 5.5: The MAP/statMAP obtained by AdapCOT (Y-axis) against the control parameter $co.AUC$.

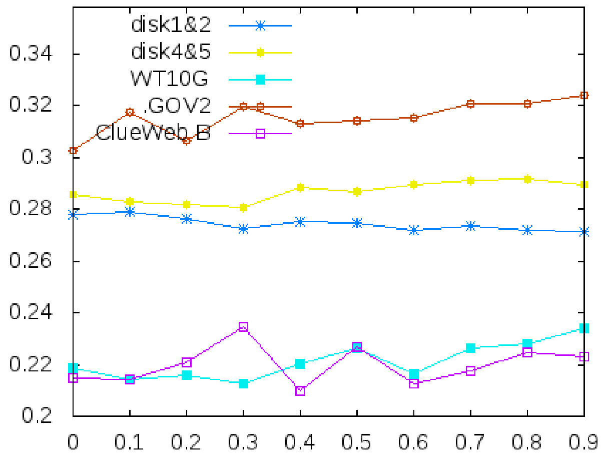
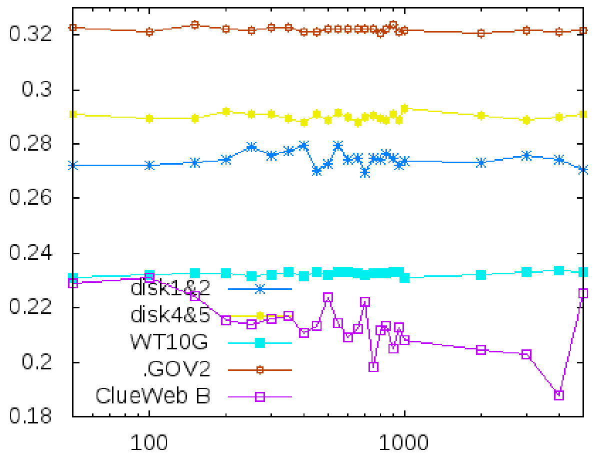


Figure 5.6: The MAP/statMAP obtained by AdapCOT (Y-axis) against the control parameter $co.maxN$.



### 5.4.3 Impact of $co.maxN$ and $co.AUC$

We also study the impact of the other two control parameters, namely the threshold ($co.AUC$) and the number of terms used for the document representation ($co.maxN$). Figures 5.5 & 5.6 plot the retrieval performance of AdapCOT against these two control parameters, respectively. It seems that the sensitivity of AdapCOT's performance to the parameters depends on the collection used. Overall, AdapCOT's parameter sensitivity is relatively high on the three Web collections, namely WT10G, .GOV2 and ClueWeb09, while being low on the other two collections. On average, a $co.maxN$ value around 100 is shown to be safe over all five collections used. On the other hand, the setting of $co.AUC$ depends on the collection used. For the three collections with relatively small sizes, namely disk1&2, disk4&5, and WT10G, the retrieval performance does not seem to be sensitive to the setting of $co.AUC$. The MAP values obtained remain stable across different $co.AUC$ settings. For the other two very large Web collections, namely .GOV2 and ClueWeb09, we observe a relatively high variance of the retrieval performance over different $co.AUC$ settings, while $co.AUC = 0.30$ leads to the best retrieval performance in average over these two very large Web collections.

### 5.4.4 Comparison with baseCOT

Table 5.6: MAP/statMAP obtained by the baseline co-training method (baseCOT) and the adaptive co-training (AdapCOT).

| Coll. | baseCOT | AdapCOT |
|---|---|---|
| disk1&2 | 0.2769 | 0.2798, 1.05% |
| disk4&5 | 0.2855 | 0.2916, 2.14% |
| WT10G | 0.2187 | 0.2338*, 6.90% |
| .GOV2 | 0.3127 | 0.3235*, 3.45% |
| ClueWeb09 | 0.2158 | 0.2330*, 7.97% |

A major advantage of our proposed approach is to employ a thresholding strategy that carries out a quality control on the learned classifiers. Therefore, we also compare our proposed AdapCOT with the original co-training method without the introduction of the threshold, which is equivalent to the algorithm described in Figure 5.3 without steps 2.(2) and 2.(8). By applying this baseline co-training method (baseCOT), the co-training process proceeds for *co.K* iterations, regardless of the quality of the learned classifiers. As shown in Table 5.6, our proposed AdapCOT method indeed improves retrieval performance over the original co-training baseline, particularly on the three Web collections. This is possibly due to the heterogeneous nature of these three Web collections, in which documents tend to be noisy, and the quality of training data becomes crucial for designing a co-training process to find good feedback documents.

Additionally, our proposed AdapCOT method does not cost more time than the QE significantly in our experiments. The AdapCOT method has extra computational cost because of the co-training process. However, this cost depends on the amount of features ($maxN$) and number of the iterations instead of the size of collections. Figure 5.6 indicates that the best results are obtained when $maxN$ is relatively small. The number of iteration in our experiments is set to 3. Since one iteration only consumes 8–10 seconds on our workstation(P4 2.6G, 4G RAM, 1.5T HDD), the extra time cost is not extensive when compared to the QE baseline.

### 5.4.5 Comparison with Term Selection Method

Table 5.7: MAP/statMAP obtained by the term selection method (TS) and AdapCOT.

| Coll. | Topics | TS | AdapCOT |
|---|---|---|---|
| AP | 51-100 | 0.3090 | 0.3251, 5.21% |
| WSJ | 51-100 | 0.3036 | 0.3111, 2.47% |
| disk4&5 | 351-400 | 0.2208 | 0.2319, 5.03% |

In this section, we compare our proposed AdapCOT with the state-of-the-art term selection method (TS) [CNGR08]. The basic idea of TS is to select expansion terms using SVM based on a list of features, including the distance of the expansion term to original query terms, the expansion term's distribution in the feedback documents, etc. In the literature, TS is the first method that applies SVM to select expansion terms. It is a strong baseline, which has shown marked (up to 28.36%) improvement over the KL-divergence language model [CNGR08].

In order to establish a fair comparison, we apply AdapCOT on the same collections with the same test queries used in [CNGR08]. Table 5.7 provides the related experimental results. Our AdapCOT method is shown to be generally more effective than TS. AdapCOT provides a moderate, up to 5.21% improvement over TS on the three test collections used. This indicates that AdapCOT's retrieval performance is at least comparable to, if not better than, the state-of-the-art TS method for enhancing QE effectiveness.

# Chapter 6

# A Hybrid Model for Ad-hoc Information Retrieval

In the past thirty years, researchers make great progress in the Information Retrieval (IR) area. A plenty of new technologies, e.g., stemming, query expansion and smoothing methods, have been introduced and help to obtain better performance in retrieving relevant documents. Some attempt to combine these technologies and prove that the strategy of combinations is very effective. But sometimes, because the overlap of different technologies, combinations do not always work. So it is important to have an ensemble model to test how a new technique can be added to existing successful combinations. However, how to make an effective combination is still largely unexplored, especially under a unified framework.

In this thesis, we propose a hybrid model to incorporate three different retrieval techniques which have proven to be effective for the ad-hoc retrieval. We analyze the best TREC systems for ad-hoc retrieval, and extend the Rocchio's feedback method by incorporating them, which are proximity, feedback document quality estimation and query performance prediction techniques, under the pseudo relevance feedback

(PRF) framework. Experimental results on various TREC datasets show that our hybrid model consistently obtains better results over the best TREC systems and how each component contributes. Because our proposed model is component-based, it is very flexible to import different techniques in the future. Meanwhile, the hybrid model can help researchers to test whether their methods are additive to improve the overall performance of ad-hoc retrieval which was mentioned in [AMWZ09].

## 6.1 A Hybrid Retrieval Model

Rocchio's algorithm [Roc71a] has been introduced in details in previous chapters.

Although the Rocchio's model has been introduced for many years, it is still effective in obtaining relevant documents. It is very flexible to adapt additional components. The traditional BM25 + Rocchio's model combination can be better. First, the query term proximity information which has proven to be useful is not considered. Second, Rocchio's algorithm views terms from different feedback documents equally. Intuitively, a candidate expansion term in a document with better quality is more likely to be relevant to the query topic. Third, the interpolation parameter $\alpha$ is always fixed across a group of queries. In fact, for a well-expressed query, the candidate feedback documents are always more reliable for relevance feedback. In this work, we use a regression model to predict this interpolation parameter. To alleviate the influence of these problems, we extend Rocchio's algorithm which refines the query representation as follows.

$$Q_1 = \alpha * (\beta * Q_0 + (1 - \beta) * Q_p) + (1 - \alpha) * \sum_{r \in R} \frac{r * q(d_r)}{|R|} \tag{6.1}$$

where $\beta$ controls how much we rely on the query term proximity information [vR77], $\alpha$ controls how much we rely on the original query, $Q_p$ is an n-gram of original query terms and $q(d_r)$ is the quality score of document $d$.

As we can see from Equation 6.1, our proposed algorithm is very flexible and can evaluate different techniques. In this work, we adopt the co-occurrence interpretation of term proximity to compute $Q_p$, where the proximity among query terms is represented by the n-gram frequencies and BM25 is used as the weighting model [HHZ11b].

Full dependencies of query terms are taken into account. For the document quality factor $q(d_r)$, we simply use the scores from the first-pass retrieval for an approximation as in [YHHL10b]. For the prediction to $\alpha$, we use the same features as in [LZ09a] to train the regression model. The difference is that we do it within Rocchio's framework and use BM25 as the basic model.

## 6.2 Experiments and Analysis

We conduct experiments on three representative test collections: disk1&2, disk4&5, and GOV2, which are used in 9 TREC years. We present the results for each TREC year such that we can directly compare our results with the best TREC systems. Detailed information about the TREC datasets and the evaluation criteria, please refer to http://trec.nist.gov. For the preprocessing of the collections, we use the Porter Stemmer and a stopword list. In addition, we only use the *title* part of the topics to retrieve.

In our experiments, we first empirically evaluate different combinations of our implemented component techniques, then evaluate how these techniques perform when they are integrated into our hybrid model.

We set the parameters to fixed values in a parsimonious way such that each component technique gets considerable improvements on most collections. In other words, there is still room for improvement if the parameters are tuned on a collection-by-collection basis. Particularly, for the basic retrieval model, we use the Okapi BM25 model, and set $b$ in BM25 to 0.3. When only the query term proximity technique is added, denoted as "BM25+Prox", we set $\beta$ to 0.3. In addition, when query expansion and document quality estimation techniques are added, denoted as "Hybrid-Fixed", we empirically set $\alpha$, $|R|$ to 0.5 and 30. However, when the query performance prediction technique is used, denoted as "Hybrid-Reg", $\alpha$ is not fixed. But it is obtained from a regression model that is trained as in [LZ09a]. When evaluating our hybrid model for a particular TREC year, the queries in the remainder TREC years

Table 6.1: Direct comparison with the best MAP results in each TREC year. In the Hybrid (Fixed) method, proximity and feedback document quality are utilized. In the Hybrid (Regression) method, all the three techniques are adapted. A "*" indicates a statistically significant improvement when a component technique is added in our algorithm.

| Method | TREC1 | TREC2 | TREC3 | TREC6 | TREC7 | TREC8 | TREC2004 | TREC2005 | TREC2006 |
|---|---|---|---|---|---|---|---|---|---|
| BM25 | 0.2292 | 0.2058 | 0.2787 | 0.2397 | 0.1819 | 0.2471 | 0.2672 | 0.3403 | 0.2965 |
| BM25+Prox | 0.2461* | 0.2111* | 0.2929* | 0.2507* | 0.1936* | 0.2582* | 0.3148* | 0.3661* | 0.3459* |
| Hybrid-Fixed | 0.2938* | 0.2913* | 0.3811* | 0.2763* | 0.2576* | 0.2909* | 0.3375* | 0.4083* | 0.3944* |
| Hybrid-Reg | 0.3012 | 0.2971 | 0.3912* | 0.2886* | 0.2611 | 0.3103* | 0.3431* | 0.4193* | 0.3921 |
| BEST TREC | 0.2062 | 0.2475 | 0.3231 | 0.2876 | 0.2614 | 0.3063 | 0.3052 | 0.4056 | 0.3737 |

on the same collection are used as training data.

From Table 6.1, we can see that our hybrid model with different component techniques can significantly outperform the basic retrieval model, which reconfirms the effectiveness of these techniques. Also, when all these component techniques are used in our hybrid model, the retrieval performance can be further improved. It indicates that performance gains from these two component techniques can be added up in our proposed hybrid model. However, when we use a regression component to predict $\alpha$, the performance gain is not very obvious compared with other parts. We conjecture the main reason is as follows: when the feedback document set is more reliable for relevance feedback, the regression component is less useful.

When compared with the best TREC systems, our proposed model obviously outperforms the best TREC systems on most collections . It is of note that the results in our work are obtained in a uniform setting across all collections while the best TREC results were from different participants independently. We believe the significant improvement is mainly from our successful integration of various IR techniques in the proposed model. In addition, according to Armstrong et al.'s survey, very few published results are better than the best TREC systems, mostly below medium systems. Our proposed model is promising, which provides an excellent avenue for future IR research, especially for evaluating the overall performance of a system (not a particular component of a system). Additionally, even if we don't use the query performance prediction technique, the results are comparable with best TREC performance. Thus, it is easy to implement our approach and obtain a good baseline in a short time.

# Chapter 7

# Summary

In this thesis, we propose four methods to improve the performance of the traditional probabilistic PRF framework. Besides the hybrid framework introduced in Chapter 6, all the other three methods are proposed to discover semantic associations among terms/documents. In this chapter, we will summarize their features and make a comparison among them.

The PRoc models which based on the proximity information is very straightforward. Because the distances among terms are already fixed in the documents, how to convert these distances into values and use them to measure the semantic associations is crucial. We propose three methods based on different ideas. Although PRoc3 is better than the other two methods, their performance is generally close. On the one hand, this indicates that the contribution of the proximity information is very stable. On the other hand, according to our experimental results, it is hard to improve the work with different conversions due to the small differences in performance. This phenomenon is reasonable because the proximity information is simply and stable. Different conversions only adjust its impact in the whole framework of the Rocchio's model. So PRoc models are simple and stable, and hard to be extended. PRoc3 is better than the state-of-the-art PRM model over the P@n metric. This indicates that terms which are closer to query terms are excellent features to improve the ranks of relevant documents. Additionally, because the calculations are very easy and all the features needed (e.g. term frequencies, positions in a document) can be read directly from the index, the PRoc models are very efficient and their time cost is almost the same as the original Rocchio's model.

Comparatively, the TopPRF framework and its three derivative models are closer to the concept "semantic". Traditional IR researchers focus on how to improve term matching because terms are very accurate information. Usually, documents contain all the query terms are very likely to be relevant, especially when all these terms are close to it. This can be considered as a fine-grained matching because the information contained in each term is accurate and stable. However, as we mentioned in Section 1, the target information for a particular query can be represented in different forms, e.g., with synonyms or abbreviations. Very fine-grained matchings neglect other associations like topic similarities among texts. Those associations can also be useful. Actually, feedback terms for PRF are terms which are very relevant to the target information and not exactly the same as the query terms. Topics are particular distributions of all the vocabulary terms. The associations over topics can be somewhat coarse-grained and different from term matchings, which can also bring something new and useful. The TopPRF provide a new idea of applying topics in PRF. It also avoids the fluctuation of performance by the impact of topic numbers for topic modeling. We can see it is an advanced step beyond the proximity-based work. Furthermore, topics can be obtained through different ways and entirely different. Based on our current progress, the work can be easily extended and achieve promising results. Because it depends on the particular topics, its performance is not so stable as PRoc models. But its performance is more consistent and even better than the state-of-the-art Topk_LDA method. The work is extremely attractive because it is very extensible. Owing to the process of generating topics for each feedback circle, the proposed methods are not so efficient. However, its time is still acceptable due to the small size of documents (feedback document) for topic modeling.

The AdapCOT method is special for its adaptive idea. Due to the "pseudo" feature, it is hard to have enough training data for machine learning methods in IR. Meanwhile, the effectiveness of classifiers should also be taken into account. AdapCOT solves the two problems at the same time. The proposed method contribute a new way of applying machine learning techniques in IR, particularly in PRF. Comparatively, the selection of classifiers is not very important. Experimental results show that different combinations of classifiers do not obtain extremely different performance. The performance of AdapCOT is very stable, too. Also, because it is based on a learning algorithm, we do not have to consider what kind of formulas should work. There are two aspects we need to concern. One is the feature split, and the other one is the evaluation method for the adaptive halt condition. It has the similar efficiency issue as the TopPRF work. The learning process will increase the time cost of PRF, but that is still acceptable.

It is worth mentioning that all the above methods do not import additional parameters. In previous work, it is common to add free parameters for interpolations. These parameters can help the models to fit the data better, but do lead to more costs for model training and validation. With two more parameters, the training process will be 100 times longer. These also the main obstacle for our hybrid models. Meanwhile, the scales of different components (e.g., different weights) are difficult to adjust. To our pragmatic experiences, even different normalizations can conduct the significantly different performance. Therefore, it is better to avoid extra parameters.

In general, we have proven three kinds of ideas which work well on improving the performance of traditional PRF. Although this is a classic area and has been researched for years, there is still much work we can do, especially via adapting semantic associations among texts. With the development of new technologies, more and more associations will be discovered and applied.

# Chapter 8

# Conclusions and Future Work

Keyword based queries cannot express the full search intents of users in many cases. How to expand the original queries effectively is a very challenging problem. Researchers have attempted to make a better query through query expansion (QE) for many years and made significant improvements. In this thesis, we figure out what kind of terms are semantically associated with the original query and how to find them. According to our work and extensive experimental results, we have discovered that proximity, topics, document quality can significantly improve the performance of traditional PRF. Machine learning methods like co-training can benefit the selection of feedback documents and make the performance better. Furthermore, all the previous work can be potentially integrated into one framework, which is our ultimate goal. With the proposed framework, it possible to make the original queries better with the latest research achievements. Generally, there are still many sub-problems deserve careful investigations to model the semantic associations in corpora. In this chapter, we make some conclusions based on our work so that they can help other researchers.

## 8.1 PRoc Methods

In Chapter 3, the novel feedback model `PRoc` is proposed by incorporating proximity information into the classic Rocchio's model. Specifically, we model the statistics of expansion terms and their proximity relationship with query terms by introducing a new concept $ptf$. We have tried three proximity measures, namely window-based method, kernel-based method and the HAL method for evaluating the relationship between expansion terms and query terms. The corresponding PRoc models based on these measures, `PRoc1`, `PRoc2` and `PRoc3`, are evaluated extensively on four standard TREC collections. In general, PRoc is very effective and outperforms the state-of-the-art feedback models in different frameworks. Comparing the three variants of PRoc, PRoc3 is more effective and robust than PRoc1 and PRoc2. So different measures do conduct different performance. Because the proximity information is straightforward, to discover a measure which is much better than others is not easy. Another important issue is about $wSize$. To find an optimal $wSize$, we can follow this rule: if a collection has plenty of documents or its average document length is large (e.g., more than 700), it is always good for us to start from 20 or smaller. Otherwise, we can try a larger starting value like 50 or more.

## 8.2 TopPRF

In Chapter 4, some useful conclusions are drawn for our work on applying topic-document information effectively in PRF as well. First of all, we investigate the "fuzzy topic" obstacle and provide evidence to justify how it affects the application of topics significantly, especially for methods relying on particular topics. Because topic modeling and the usage of topics are more and more popular, this problem is critical and cannot be ignored. To this end, we propose a new probabilistic framework TopPRF by introducing a new concept topic space. Using topic space coordinates to describe documents and to compare them with complete topic-document information can bring very stable results. To identify which documents are more reliable than others, we need to weight the feedback documents by integrating topical information. Using our methods, the relative ranks of feedback documents according to their scores (e.g., cosine similarity scores) are very robust. No matter how the topic distributions change, terms in those highly ranked documents will be consistently more important than others for query expansion. It is an important finding for integrating topical information for IR, especially when there is not an optimal topic number for corpora. By using all topical information in the feedback documents, our proposed approaches have more advantage for further improving strong basic models.

Secondly, based on the new framework, we find that topic similarity is effective for evaluating the reliability of each feedback document on their relevance. However, different similarity functions will lead different performance. For instance, TS-COS performs better than TS-EU in most cases. Thirdly, because TopPRF is derived from the Rocchio's model when the performance of the latter is not good, our proposed methods are affected. The average performance of TS-COS is better than the Rocchio's model. Therefore, how to transfer the information of a document in the topic space into weights needs a very careful consideration. Fourthly, the TS-Entropy performs not so good as TS-COS or TS-EU. But it obtains better results than the baselines for most cases. So "purity" should be a useful feature when most feedback documents are really relevant. It can also be considered as a useful feature when measuring the quality of a document for other applications. Finally, when the P@n performance of the basic model is good enough, e.g., above 0.5, the size of the trustable group will not affect much. When most documents in the trustable group are irrelevant, or the size of the collection is large, the performance of our proposed methods will drop significantly when the group contains more than ten documents. By default, 3 is a good choice for different collections. This result also verifies the assumption that more than 1 document is needed to cover the related topics for a query. In summary, the “fuzzy topic” problem deserves more concern and the usage of “topic space” will be a promising solution for further applications of topics.

## 8.3 AdapCOT

In Chapter 5, the work of AdapCOT indicates the necessity to distillate good feedback documents for QE. A good feedback document should have a general interest in the query topic throughout itself, while a bad feedback document, even if it is relevant, may have only a passing-by interest in the query topic, and contains many off-topic terms. Therefore, there is a need for the distillation of the high-quality feedback documents to improve QE's effectiveness and robustness. To address this problem, we have proposed an adaptive co-training method, called AdapCOT, to find good feedback documents for QE. The proposed method overcomes a major problem of applying machine learning methods to QE, namely the lack of proper training data. On five TREC collections, extensive experiments confirm our argument that QE's retrieval performance can be improved by selecting high-quality feedback documents. According to the experimental results over five standard TREC test collections, our proposed AdapCOT leads to considerable and statistically significant improvement over the QE baseline. In particular, AdapCOT is very effective on the large-scale Web collections, showing that our proposed method can be applied in a Web environment. Moreover, current IR systems can benefit from our proposed method by a good choice of feedback documents, since most of them offer the functionality of QE in the form of feedback based on a pseudo relevance set.

## 8.4 Hybrid Framework

Finally, in Chapter 6, we provide a convenient way to achieve a strong baseline. In this study, we have discovered three kinds of technique which can be effectively integrated into the Rocchio's model, and each of them can contribute to the overall performance independently. The performance of the proposed hybrid model even surpasses the best performance in 9 TREC years. Applying proximity into the basic model BM25, utilizing document quality and estimating the parameter $\alpha$ are all very useful. It is simple for other researchers to adapt their work into this hybrid model and evaluate how their studies can make improvements over our work.

## 8.5 Future Work

While the work on utilizing topics in PRF is just a start, we have many ideas for future extensions. Firstly, we plan to research on different similarity formulas which can affect the performance significantly. Since there are many choices, it is promising to have better results based on the topic similarity. Secondly, we can combine the topic similarity feature with topic-entropy or other features to investigate how to integrate them effectively. Thirdly, with the development of topic modeling, we can try more topic models to see how topics obtained through them influence our methods and why that happens. Finally, the conclusions in the thesis can be used for other text process areas to discover high-quality documents or measure document similarity. Besides topics, with the development of semantic web, natural language processing, knowledge graphs and other related areas, more and more semantic associations can be discovered and applied in IR.

For the AdapCOT method, we can employ a co-training process that makes use of two nature sets of document features, namely the term-based features [HHW$^+$06a] and the high-level features [HO09a, CNGR08]. To extend the work on applying proximity information in PRF, possible research direction is to find the exact relationship between the window size factor and the information of collections, e.g., the length distribution of documents. It is also interesting to apply our work to other retrieval frameworks, like DFR or the language modeling framework.

We will continue our work on the hybrid PRF framework. It will be interesting to integrate our studies in it and evaluate their impacts on the current hybrid model. While more and more techniques are developed, it is meaningful and promising to categorize them and adapt them into the unified framework. The community will be benefit from this because researchers can make accumulated progress on the foundation of previous research.

Natural language processing and knowledge graph technologies are closer to the concept “semantic” than methods in this thesis. They are not suitable for traditional IR for some reasons. First of all, most indices are still based on terms so that the structures of corpora are destroyed. Syntactic methods cannot be utilized in this case. Secondly, ontologies and knowledge graphs are not complete and accurate in the past. Also, not all relations are useful for IR. Finally, both kinds of technologies require strong computation abilities and huge data. Mining useful information from the data is like seeking a needle in the sea. With the development of these technologies, we believe it is promising to apply them in IR in the future.

# Bibliography

[AB11] David Andrzejewski and David Buttler. Latent topic feedback for information retrieval. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 600–608, New York, NY, USA, 2011. ACM.

[ABA+01] Claudio Carpineto A, Renato De Mori B, Giovanni Romano A, Brigitte Bigi B, and A Fondazione Ugo Bordoni. An information-theoretic approach to automatic query expansion. *ACM Trans. Inf. Syst.*, 19:1–27, January 2001.

[ACC+00] James Allan, Margaret E. Connell, W. Bruce Croft, Fangfang Feng, David Fisher, and Xiaoyan Li. INQUERY and TREC-9. In *TREC*, 2000.

[ACR04] Giambattista Amati, Claudio Carpineto, and Giovanni Romano. Query difficulty, robustness, and selective application of query expansion. In Sharon McDonald and John Tait, editors, *ECIR*, volume 2997 of *Lecture Notes in Computer Science*, pages 127–137. Springer, 2004.

[Ama03a] G. Amati. *Probabilistic models for information retrieval based on divergence from randomness.* PhD thesis, Department of Computing Science, University of Glasgow, 2003.

[Ama03b] G. Amati. Probabilistic models for information retrieval based on divergence from randomness. *PhD thesis, Department of Computing Science, University of Glasgow*, 2003.

[AMWZ09] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. Improvements that don't add up: ad-hoc retrieval results since 1998. In Cheung et al. [CSC+09], pages 601–610.

[APY06] Javed A. Aslam, Virgil Pavlu, and Emine Yilmaz. A statistical method for system evaluation using incomplete judgments. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 541–548, New York, NY, USA, 2006. ACM.

[BC06] Stefan Büttcher and Charles L. A. Clarke. A document-centric approach to static index pruning in text retrieval systems. In *Proceedings of the 15th ACM International Conference on Information and Knowledge*

*Management*, CIKM '06, pages 182–189, New York, NY, USA, 2006. ACM.

[BCL06] 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.

[BM98] Avrim Blum and Tom M. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100, 1998.

[BNJ03a] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[BNJ03b] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[Buc94] Chris Buckley. Automatic query expansion using smart : Trec 3. In *In Proceedings of The third Text REtrieval Conference (TREC-3*, pages 69–80, 1994.

[CA12] Karla L. Caballero and Ram Akella. Incorporating statistical topic information in relevance feedback. In *Proceedings of the 35th Interna-*

*tional ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 1093–1094, New York, NY, USA, 2012. ACM.

[CCB01] G. Romano C. Carpineto, R. de Mori and B. Bigi. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems (TOIS)*, 19(1):1–27, 2001.

[CCT00] Charles L.A. Clarke, Gordon V. Cormack, and Elizabeth A. Tudhope. Relevance ranking for one to three term queries. *Information Processing Management*, 36(2):291 – 311, 2000.

[CDF$^+$00] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom M. Mitchell, Kamal Nigam, and Seán Slattery. Learning to construct knowledge bases from the world wide web. *Artif. Intell.*, 118(1-2):69–113, 2000.

[CKC$^+$08] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st SIGIR*, pages 659–666. ACM, 2008.

[CKP04] Jason Chan, Irena Koprinska, and Josiah Poon. Co-training with a single natural feature set applied to email classification. In *Web Intelligence*, pages 586–589. IEEE Computer Society, 2004.

[CMSS14] J Shane Culpepper, Stefano Mizzaro, Mark Sanderson, and Falk Scholer. Trec: Topic engineering exercise. In *Proceedings of the 37th SIGIR*, SIGIR '14, pages 1147–1150, New York, NY, USA, 2014. ACM.

[CNGR08] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *SIGIR*, pages 243–250. ACM, 2008.

[CPL15] Ronan Cummins, Jiaul H. Paik, and Yuanhua Lv. A pólya urn document language model for improved information retrieval. *ACM Trans. Inf. Syst.*, 33(4):21:1–21:34, May 2015.

[CR12] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1):1:1–1:50, January 2012.

[CRG02] Claudio Carpineto, Giovanni Romano, and Vittorio Giannini. Improving retrieval feedback with multiple term-ranking function combination.

*ACM Trans. Inf. Syst.*, 20(3):259–290, 2002.

[CSC+09] David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy J. Lin, editors. *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009.* ACM, 2009.

[CT09] Kevyn Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 837–846, New York, NY, USA, 2009. ACM.

[DM06] Fernando Diaz and Donald Metzler. Improving the estimation of relevance models using large external corpora. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, New York, NY, USA, 2006. ACM.

[Eft96] Efthimis N. Efthimiadis. Query expansion. *Annual review of information systems and technology*, 31, 1996.

[GG84] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern*

*Anal. Mach. Intell.*, 6(6):721–741, 1984.

[GGZ$^+$04] David A. Grossman, Luis Gravano, ChengXiang Zhai, Otthein Herzog, and David A. Evans, editors. *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, USA, November 8-13, 2004*. ACM, 2004.

[GS04] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[GT04] DMBTL Griffiths and MIJJB Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16:17, 2004.

[GZ00] Sally A. Goldman and Yan Zhou. Enhancing supervised learning with unlabeled data. In Pat Langley, editor, *ICML*, pages 327–334. Morgan Kaufmann, 2000.

[Hay95] Carla Hayn. The information content of losses. *Journal of Accounting and Economics*, 20(2):125 – 153, 1995.

[HHW$^+$06a] Xiangji Huang, Yan Rui Huang, Miao Wen, Aijun An, Yang Liu, and Josiah Poon. Applying data mining to pseudo-relevance feedback for high performance text retrieval. In *ICDM*, pages 295–306. IEEE Computer Society, 2006.

[HHW+06b] Xiangji Huang, Yan Rui Huang, Miao Wen, Aijun An, Yang Liu, and Josiah Poon. Applying data mining to pseudo-relevance feedback for high performance text retrieval. In *Proceedings of the 6th ICDM*, pages 295–306. IEEE, 2006.

[HHZ11a] Ben He, Jimmy Xiangji Huang, and Xiaofeng Zhou. Modeling term proximity for probabilistic information retrieval models. *Information Sciences*, 181(14):3017 – 3031, 2011.

[HHZ11b] Ben He, Jimmy Xiangji Huang, and Xiaofeng Zhou. Modeling term proximity for probabilistic information retrieval models. *Inf. Sci.*, 181(14):3017–3031, 2011.

[HMH13] Jimmy Xiangji Huang, Jun Miao, and Ben He. High performance query expansion using adaptive co-training. *Information Processing & Management*, 49(2):441–453, 2013.

[HO09a] Ben He and Iadh Ounis. Finding good feedback documents. In Cheung et al. [CSC+09], pages 2011–2014.

[HO09b] Ben He and Iadh Ounis. Studying query expansion effectiveness. In Mohand Boughanem, Catherine Berrut, Josiane Mothe, and Chantal Soulé-Dupuy, editors, *ECIR*, volume 5478 of *Lecture Notes in Computer Science*, pages 611–619. Springer, 2009.

[Hof99] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd SIGIR*, pages 50–57, New York, NY, USA, 1999. ACM.

[Joa99] Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *ICML*, pages 200–209. Morgan Kaufmann, 1999.

[Kee91] E. Michael Keen. The use of term position devices in ranked output experiments. *J. Doc.*, 47:1–22, 1991.

[Kee92] E. Michael Keen. Some aspects of proximity searching in text retrieval systems. *J. Inf. Sci.*, 18:89–98, 1992.

[KGV83] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598), 1983.

[KL95] Ruth Ann Atchley Kevin Lund, Curt Burgess. Semantic and associative priming in high-dimensional semantic space. In *Proceedings of the 17th CogSci*, pages 660–665, 1995.

[KMAH04] Svetlana Kiritchenko, Stan Matwin, and Suhayya Abu-Hakima. Email classification with temporal features. In Mieczyslaw A. Klopotek, Slawomir T. Wierzchon, and Krzysztof Trojanowski, editors, *Intelligent Information Systems*, Advances in Soft Computing, pages 523–533. Springer, 2004.

[KSCC11] Mostafa Keikha, Jangwon Seo, W Bruce Croft, and Fabio Crestani. Predicting document effectiveness in pseudo relevance feedback. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2061–2064. ACM, 2011.

[LBCC04] Thomas R. Lynam, Chris Buckley, Charles L. A. Clarke, and Gordon V. Cormack. A multi-system analysis of document and term selection for blind feedback. In Grossman et al. [GGZ$^+$04], pages 261–269.

[LC01a] 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, USA, 2001. ACM.

[LC01b] Victor Lavrenko and W. Bruce Croft. Relevance-based language models. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin

Zobel, editors, *SIGIR*, pages 120–127. ACM, 2001.

[LCA08] Kyung Soon Lee, W. Bruce Croft, and James Allan. A cluster-based resampling method for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 235–242, New York, NY, USA, 2008. ACM.

[LM06] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 577–584, New York, NY, USA, 2006. ACM.

[LZ09a] Yuanhua Lv and ChengXiang Zhai. Adaptive relevance feedback in information retrieval. In Cheung et al. [CSC$^{+}$09], pages 255–264.

[LZ09b] 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.

[LZ09c] Yuanhua Lv and ChengXiang Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In Cheung et al. [CSC+09], pages 1895–1898.

[LZ09d] Yuanhua Lv and ChengXiang Zhai. Positional language models for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 299–306, New York, NY, USA, 2009. ACM.

[LZ10] Yuanhua Lv and ChengXiang Zhai. Positional relevance model for pseudo-relevance feedback. In *Proceeding of the 33rd SIGIR conference*, SIGIR '10, pages 579–586. ACM, 2010.

[MC05] Donald Metzler and W. Bruce Croft. A markov random field model for term dependencies. In *SIGIR '05: Proceedings of the 28th annual International ACM SIGIR Conference on Research and Development in information retrieval*, pages 472–479, New York, NY, USA, 2005. ACM.

[MC07] Donald Metzler and W. Bruce Croft. Latent concept expansion using markov random fields. In *SIGIR '07: Proceedings of the 30th annual* *international ACM SIGIR conference on Research and development in information retrieval*, pages 311–318, New York, NY, USA, 2007. ACM.

[MHH10] Jun Miao, Xiangji Huang, and Qinmin Hu. York_university at TRECVID 2010. In *TRECVID 2010 workshop participants notebook papers, Gaithersburg, MD, USA, November 2010*, 2010.

[MHY12] Jun Miao, Jimmy Xiangji Huang, and Zheng Ye. Proximity-based rocchio's model for pseudo relevance. In *Proceedings of the 35th SIGIR*, SIGIR '12, pages 535–544, New York, NY, USA, 2012. ACM.

[MHZ16] Jun Miao, Jimmy Xiangji Huang, and Jiashu Zhao. Topprf: A probabilistic framework for integrating topic space into pseudo relevance feedback. *ACM Trans. Inf. Syst.*, 34(4):22:1–22:36, August 2016.

[MNCR09] Donald Metzler, Jasmine Novak, Hang Cui, and Srihari Reddy. Building enriched document representations using aggregated anchor text. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 219–226, New York, NY, USA, 2009. ACM.

[MSB98] Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In *SIGIR*, pages 206–214. ACM, 1998.

[MSZ07] Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 490–499, New York, NY, USA, 2007. ACM.

[NMTM00] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2/3):103–134, 2000.

[Pla99] John C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999.

[PNI$^+$08] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD*, pages 569–577. ACM, 2008.

[PO07] Vassilis Plachouras and Iadh Ounis. Multinomial randomness models for retrieval with document fields. In *Proceedings of ECIR*, pages 28–39, 2007.

[Por80] M. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[Ram03] Juan Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, 2003.

[RFK02] Bhavani Raskutti, Herman L. Ferrá, and Adam Kowalczyk. Combining clustering and co-training to enhance text classification using unlabelled data. In *KDD*, pages 620–625. ACM, 2002.

[Roc71a] J. Rocchio. Relevance feedback in information retrieval. pages 313–323. Prentice-Hall Englewood Cliffs, 1971.

[Roc71b] J. J. Rocchio. Relevance feedback in information retrieval. In *G. Salton, The SMART retrieval system: Experiments in automatic document*, pages 313–323, 1971.

[RS03] Yves Rasolofo and Jacques Savoy. Term proximity scoring for keyword-based retrieval systems. In Fabrizio Sebastiani, editor, *Advances in Information Retrieval*, volume 2633 of *Lecture Notes in Computer Science*, pages 79–79. Springer Berlin / Heidelberg, 2003.

[RWHB+95] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Mike Gatford, and A. Payne. Okapi at TREC-4. In *TREC*, 1995.

[RWJ$^+$94a] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, pages 109–126, 1994.

[RWJ$^+$94b] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *TREC*, pages 109–126, 1994.

[RZ09] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond.* Now Publishers Inc, 2009.

[RZT04] Stephen E. Robertson, Hugo Zaragoza, and Michael J. Taylor. Simple BM25 extension to multiple weighted fields. In Grossman et al. [GGZ$^+$04], pages 42–49.

[Sal71] Gerald Salton. *The SMART Retrieval System.* Prentice Hall, New Jersey, 1971.

[SK13] Midori Serizawa and Ichiro Kobayashi. A study on query expansion based on topic distributions of retrieved documents. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Process-*

*ing*, volume 7817 of *Lecture Notes in Computer Science*, pages 369–379. Springer Berlin Heidelberg, 2013.

[STW$^+$08] Ruihua Song, Michael Taylor, Ji-Rong Wen, Hsiao-Wuen Hon, and Yong Yu. Viewing term proximity from a different perspective. In Craig Macdonald, Iadh Ounis, Vassilis Plachouras, Ian Ruthven, and Ryen White, editors, *Advances in Information Retrieval*, volume 4956 of *Lecture Notes in Computer Science*, pages 346–357. Springer Berlin / Heidelberg, 2008.

[SWY75a] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975.

[SWY75b] G. Salton, A. Wong, and C.S. Yang. A vector space model for information retrieval. *Journal of American Society for Information Retrieval*, 18(11):613–620, November 1975.

[SYY98] Henry Stark, Yongi Yang, and Yongyi Yang. *Vector space projections: a numerical approach to signal and image processing, neural nets, and optics*. John Wiley & Sons, Inc., 1998.

[TJZ08] Jie Tang, Ruoming Jin, and Jing Zhang. A topic modeling approach and its integration into the random walk framework for academic search. In

*Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 1055–1060. IEEE, 2008.

[TZ06] Tao Tao and ChengXiang Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 162–169, New York, NY, USA, 2006. ACM.

[TZ07] 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, USA, 2007. ACM.

[VH00] Ellen M. Voorhees and Donna Harman. Overview of the sixth text retrieval conference. *Information Processing and Management: an International Journal*, 36:3–35, July 2000.

[Voo05] Ellen Voorhees. *TREC: Experiment and Evaluation in Information Retrieval*. The MIT Press, 2005.

[vR77] C. J. van Rijsbergen. A theoretical basis for the use of co-occurence data in information retrieval. *Journal of Documentation*, 1977.

[VW06] 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.

[Wal04] B. Walsh. Markov chain monte carlo and gibbs sampling, 2004.

[WB11] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 448–456, New York, NY, USA, 2011. ACM.

[WC06] 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*, pages 178–185. ACM, 2006.

[WF05] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, San Francisco, 2005.

[WZWS12] Xuwen Wang, Qiang Zhang, Xiaojie Wang, and Yueping Sun. LDA based pseudo relevance feedback for cross language information retrieval. In *Cloud Computing and Intelligent Systems (CCIS)*, volume 03, pages 1511–1516, Oct 2012.

[XA08] Zuobing Xu and Ram Akella. A bayesian logistic regression model for active relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 227–234, New York, NY, USA, 2008. ACM.

[XC96] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *SIGIR*, pages 4–11. ACM, 1996.

[XC00] Jinxi Xu and W. Bruce Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Trans. Inf. Syst.*, 18(1):79–112, January 2000.

[YA08] Xing Yi and James Allan. Evaluating topic models for information retrieval. In *Proceedings of the 17th ACM CIKM*, pages 1431–1432. ACM, 2008.

[YA09] Xing Yi and James Allan. A comparative study of utilizing topic models for information retrieval. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 29–41, Berlin, Heidelberg, 2009. Springer-Verlag.

[YH16] Zheng Ye and Jimmy Xiangji Huang. A learning to rank approach for quality-aware pseudo-relevance feedback. *Journal of the Association for Information Science and Technology*, 67(4):942–959, 2016.

[YHHL09] Zheng Ye, Xiangji Huang, Ben He, and Hongfei Lin. YorkUniversity at TREC 2009: Relevance feedback track. In *Notebook of the Eighteenth Text REtrieval Conference (TREC2009)*, Gaithersburg, Maryland, 2009.

[YHHL10a] Zheng Ye, Ben He, Xiangji Huang, and Hongfei Lin. Revisiting rocchio's relevance feedback algorithm for probabilistic models. pages 151–161. AIRS, 2010.

[YHHL10b] Zheng Ye, Ben He, Xiangji Huang, and Hongfei Lin. Revisiting rocchio's relevance feedback algorithm for probabilistic models. In Pu-Jen Cheng, Min-Yen Kan, Wai Lam, and Preslav Nakov, editors, *AAIRS*, volume 6458 of *Lecture Notes in Computer Science*, pages 151–161. Springer, 2010.

[YHL11a] Zheng Ye, Jimmy Xiangji Huang, and Hongfei Lin. Finding a good query-related topic for boosting pseudo-relevance feedback. *J. Am. Soc. Inf. Sci. Technol.*, 62(4):748–760, April 2011.

[YHL11b] Zheng Ye, Jimmy Xiangji Huang, and Hongfei Lin. Finding a good query-related topic for boosting pseudo-relevance feedback. *Journal of the American Society for Information Science and Technology*, 62(4):748–760, 2011.

[YHL11c] Zheng Ye, Xiangji Huang, and Hongfei Lin. A bayesian network approach to context sensitive query expansion. In *SAC*, pages 1138–1142, 2011.

[YHLZ13] Xiaoshi Yin, Jimmy Xiangji Huang, Zhoujun Li, and Xiaofeng Zhou. A survival modeling approach to biomedical search result diversification using wikipedia. *IEEE Trans. Knowl. Data Eng.*, 25(6):1201–1212, 2013.

[YHM12a] Zheng Ye, Jimmy Xiangji Huang, and Jun Miao. A hybrid model for ad-hoc information retrieval. In *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012*, pages 1025–1026, 2012.

[YHM12b] Zheng Ye, Jimmy Xiangji Huang, and Jun Miao. A hybrid model for ad-hoc information retrieval. In *Proceedings of the 35th International*

*ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 1025–1026, New York, NY, USA, 2012. ACM.

[ZH01] Sarah Zelikovitz and Haym Hirsh. Using LSI for text classification in the presence of background text. In *CIKM*, pages 113–118. ACM, 2001.

[Zha08a] ChengXiang Zhai. Statistical language models for information retrieval. *Synthesis Lectures on Human Language Technologies*, 1(1):1–141, 2008.

[Zha08b] ChengXiang Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2:137–213, March 2008.

[ZHH11] Jiashu Zhao, Jimmy Xiangji Huang, and Ben He. CRTER: using cross terms to enhance probabilistic information retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 155–164, New York, USA, 2011. ACM.

[ZHH$^+$12] Jiashu Zhao, Jimmy Xiangji Huang, Xiaohua Hu, C. Joseph Kurian, and William Melek. A bayesian-based prediction model for personalized medical health care. In *2012 IEEE International Conference on*

*Bioinformatics and Biomedicine, BIBM 2012, Philadelphia, PA, USA, October 4-7, 2012*, pages 1–4, 2012.

[ZHY14] Jiashu Zhao, Jimmy Xiangji Huang, and Zheng Ye. Modeling term associations for probabilistic information retrieval. *ACM Trans. Inf. Syst.*, 32(2):7, 2014.

[ZL01a] C. Zhai and J. 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*, pages 334–342, New Orleans, LA, 2001.

[ZL01b] Chengxiang Zhai and John Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 403–410. ACM, 2001.

[ZL01c] 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.

[ZL04] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.

[ZY09] Jinglei Zhao and Yeogirl Yun. A proximity language model for information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference*, SIGIR '09, pages 291–298, New York, USA, 2009. ACM.

# Appendix A

## Scores of top 20 documents on topic 802 and 804 through TS-COS

Table A.1: Top 20 document TS-COS scores given topic# 5, 10 and 30 on query 802 “Volcano eruptions global temperature” and 804 “ban on human cloning”

| Volcano eruptions global temperature | | | | | |
|---|---|---|---|---|---|
| 5 | | 10 | | 30 | |
| 1 | 1.0000 | 1 | 1.0000 | 1 | 1.0000 |
| 2 | 1.0000 | 2 | 1.0000 | 2 | 1.0000 |
| 3 | 1.0000 | 3 | 1.0000 | 3 | 1.0000 |
| 4 | 0.9091 | 4 | 0.7660 | 4 | 0.7684 |
| 5 | 0.8529 | 5 | 0.7620 | 5 | 0.7238 |
| 6 | 0.8162 | 6 | 0.6440 | 6 | 0.6108 |
| 7 | 0.7395 | 7 | 0.6778 | 7 | 0.6892 |
| 8 | 0.8519 | 8 | 0.7627 | 8 | 0.7416 |
| 9 | 0.5373 | 9 | 0.5387 | 9 | 0.5066 |
| 10 | 0.8866 | 10 | 0.7803 | 10 | 0.7359 |
| 11 | 0.8493 | 11 | 0.7630 | 11 | 0.7984 |
| 12 | 0.5371 | 12 | 0.5396 | 12 | 0.5070 |
| 13 | 0.5378 | 13 | 0.5451 | 13 | 0.5071 |
| 14 | 0.6200 | 14 | 0.6127 | 14 | 0.5762 |
| 15 | 0.5366 | 15 | 0.5445 | 15 | 0.5073 |
| 16 | 0.5399 | 16 | 0.5446 | 16 | 0.5074 |
| 17 | 0.8441 | 17 | 0.8012 | 17 | 0.7642 |
| 18 | 0.8944 | 18 | 0.8939 | 18 | 0.8184 |
| 19 | 0.8265 | 19 | 0.6337 | 19 | 0.6170 |
| 20 | 0.8358 | 20 | 0.6248 | 20 | 0.6188 |

Table A.2: Top 20 document TS-COS scores given topic# 5, 10 and 30 on 804 “ban on human cloning”

| ban on human cloning | | | | | |
|---|---|---|---|---|---|
| 5 | | 10 | | 30 | |
| 1 | 1.0000 | 1 | 1.0000 | 1 | 1.0000 |
| 2 | 1.0000 | 2 | 1.0000 | 2 | 1.0000 |
| 3 | 1.0000 | 3 | 1.0000 | 3 | 1.0000 |
| 4 | 0.8697 | 4 | 0.8763 | 4 | 0.8693 |
| 5 | 0.6556 | 5 | 0.6062 | 5 | 0.7860 |
| 6 | 0.6544 | 6 | 0.6114 | 6 | 0.7867 |
| 7 | 0.6506 | 7 | 0.5923 | 7 | 0.7998 |
| 8 | 0.8928 | 8 | 0.8795 | 8 | 0.8025 |
| 9 | 0.8601 | 9 | 0.8149 | 9 | 0.6950 |
| 10 | 0.7315 | 10 | 0.6391 | 10 | 0.7742 |
| 11 | 0.8780 | 11 | 0.6036 | 11 | 0.6921 |
| 12 | 0.6408 | 12 | 0.6224 | 12 | 0.7685 |
| 13 | 0.6414 | 13 | 0.6219 | 13 | 0.7682 |
| 14 | 0.6505 | 14 | 0.5896 | 14 | 0.7985 |
| 15 | 0.6492 | 15 | 0.5906 | 15 | 0.7941 |
| 16 | 0.8507 | 16 | 0.8583 | 16 | 0.8816 |
| 17 | 0.8603 | 17 | 0.8142 | 17 | 0.7261 |
| 18 | 0.7249 | 18 | 0.7369 | 18 | 0.8234 |
| 19 | 0.7899 | 19 | 0.7696 | 19 | 0.8627 |
| 20 | 0.8851 | 20 | 0.8706 | 20 | 0.8587 |

# Appendix B

# Author's Publication List

- Jun Miao, Jimmy Xiangji Huang: TopPRF: A Probabilistic Framework for Integrating Topic Space into Pseudo Relevance Feedback. Accepted by ACM Transactions on Information Systems (TOIS), 2016
- Jimmy Xiangji Huang, Jun Miao, Ben He: High performance query expansion using adaptive co-training. Inf. Process. Manage. 49(2): 441-453 (2013)
- Jun Miao, Jimmy Xiangji Huang, Zheng Ye: Proximity-based rocchio's model for pseudo relevance. SIGIR 2012: 535-544
- Zheng Ye, Jimmy Xiangji Huang, Jun Miao: A hybrid model for ad-hoc information retrieval. SIGIR 2012: 1025-1026
- Jun Miao, Zheng Ye, Jimmy Huang: York University at TREC 2012: Medical Records Track. TREC 2012
- Qinmin Hu, Jimmy Huang, Jun Miao: A robust approach to optimizing multi-source information for enhancing genomics retrieval performance. BMC Bioin-

formatics 12(S-5): S6 (2011)

- Mariam Daoud, Dawid Kasperowicz, Jun Miao, Jimmy Huang: York University at TREC 2011: Medical Records Track. TREC 2011

- Qinmin Vivian Hu, Jimmy Xiangji Huang, Jun Miao: Exploring a multi-source fusion approach for genomics information retrieval. BIBM 2010: 669-672

- Jun Miao, Xiangji Huang, Qinmin Hu: York University at TRECVID 2010. TRECVID 2010

# Appendix C

# Sample of Topics for Datasets

## C.1 Disk 1&2

<top>

<head> Tipster Topic Description

<num> Number: 101

<dom> Domain: Science and Technology

<title> Design of the Star Wars Anti missile Defense System

<desc> Description:

Document will provide information on the proposed configuration, components, and technology of the U.S. s star wars anti missile defense system.

<smry> Summary:

Document will provide information on the proposed configuration, components, and technology of the U.S. s star wars anti missile defense system.

<narr> Narrative:

A relevant document will provide information which aids description of the design and technology to be used in the anti missile defense system advocated by the Reagan administration, the Strategic Defense Initiative (SDI), also known as star wars. Any reported changes to original design, or any research results which might lead to changes of constituent technologies, are also relevant documents. However, reports on political debate over the SDI, or arms control negotiations which might encompass the SDI, are NOT relevant to the science and technology focus of this topic, unless they provide specific information on design and technology.

<con> Concept(s):

1. Strategic Defense Initiative, SDI, star wars, peace shield

2. kinetic energy weapon, kinetic kill, directed energy weapon, laser, particle beam, ERIS (exoatmospheric reentry vehicle interceptor system), phased array radar, microwave

3. anti satellite (ASAT) weapon, spaced-based technology, strategic defense technologies

provide specific information on design and technology.

<con> Concept(s):

1. Strategic Defense Initiative, SDI, star wars, peace shield

2. kinetic energy weapon, kinetic kill, directed energy weapon, laser, particle beam, ERIS (exoatmospheric reentry vehicle interceptor system), phased array radar,

microwave

3. anti satellite (ASAT) weapon, spaced-based technology, strategic defense technologies

<fac> Factor(s):

<nat> Nationality: U.S.

</nat>

<def> Definition(s):

</top>

## C.2 Disk4&5

<top>

<num> Number: 302 <title> Poliomyelitis and Post-Polio

<desc> Description: Is the disease of Poliomyelitis (polio) under control in the world?

<narr> Narrative: Relevant documents should contain data or outbreaks of the polio disease (large or small scale), medical protection against the disease, reports on what has been labeled as "post-polio" problems. Of interest would be location of the cases, how severe, as well as what is being done in the "post-polio" area.

</top>

## C.3 WT2G

<top>

<num> Number: 401 <title> foreign minorities, Germany

<desc> Description: What language and cultural differences impede the integration of foreign minorities in Germany?

<narr> Narrative: A relevant document will focus on the causes of the lack of integration in a significant way; that is, the mere mention of immigration difficulties is not relevant. Documents that discuss immigration problems unrelated to Germany are also not relevant.

</top>

## C.4 WT10G

<top>

<num> Number: 451 <title> What is a Bengals cat

<desc> Description: Provide information on the Bengal cat breed.

<narr> Narrative: Item should include any information on the Bengal cat breed, including description, origin, characteristics, breeding program, names of breeders and catteries carrying bengals. References which discuss bengal clubs only are not

relevant. Discussions of bengal tigers are not relevant.

</top>

## C.5 GOV2

<top> <num> Number: 701

<title> U.S. oil industry history

<desc> Description: Describe the history of the U.S. oil industry

<narr> Narrative: Relevant documents will include those on historical exploration and drilling as well as history of regulatory bodies. Relevant are history of the oil industry in various states, even if drilling began in 1950 or later.

</top>

## C.6 ClueWeb09

<topic number="1" type="faceted">

<query>obama family tree</query>

<description>Find information on President Barack Obama's family history, including genealogy, national origins, places and dates of birth, etc. </description>

<subtopic number="1" type="nav"> Find the TIME magazine photo essay "Barack Obama's Family Tree". </subtopic>

<subtopic number=”2” type=”inf”> Where did Barack Obama’s parents and grandparents come from? </subtopic>

<subtopic number=”3” type=”inf”> Find biographical information on Barack Obama’s mother. </subtopic>

</topic>

# Appendix D

## Related Java code for the Proximity work

Because different researchers use different IR systems for their work, so here we just show some related code for the different proximity methods. We only use part of them and wish people can use all of them as distance calculators.

```
\footnotesize
import java.util.Arrays;

import org.apache.commons.math.distribution.NormalDistributionImpl;
import org.apache.log4j.Logger;

public class Distance {
        private static Logger logger = Logger.getLogger(Distance.class);

        protected static double sigmoidPower = Double.parseDouble(ApplicationSetup
                        .getProperty("sigmoid.power", "1d"));




        /**
         *
         * @param positionOfTerm1
         * @param positionOfTerm2
         * @return
         */
        public static double unorderGaussianTimes(final int[] positionOfTerm1,
                        final int[] positionOfTerm2, NormalDistributionImpl nDist) {
                double retValue =0;
                if (positionOfTerm1 == null || positionOfTerm2 == null) {
                        return 0;
                }
                for(int i=0; i < positionOfTerm1.length; i++){
                        for(int j =0; j < positionOfTerm2.length; j++){
                                double dist = nDist.density(Math.abs(positionOfTerm1[i] -
                                     positionOfTerm2[j]));
                                if(dist > 0){
                                        retValue += dist;
                                }
                        }
                }
                return retValue;
        }
        /**
         *
         * @param positionOfTerm1
         * @param positionOfTerm2
         * @param winSize can exceed the size the the documents
         * @return
         */
        public static double unorderGaussianTimes(final int[] positionOfTerm1,
                        final int[] positionOfTerm2, final int winSize, NormalDistributionImpl
                            nDist) {
                double retValue =0;
                if (positionOfTerm1 == null || positionOfTerm2 == null) {
                        return 0;
                }
                for(int i=0; i < positionOfTerm1.length; i++){
                        for(int j =0; j < positionOfTerm2.length; j++){
                                double tmp = Math.abs(positionOfTerm1[i] - positionOfTerm2[j]);
                                double dist = nDist.density(tmp);
                                if(dist > 0 && tmp < winSize){
                                        retValue += dist;
                                }else{
//                                              System.out.println(tmp +":" + dist);
                                }


                        }
                }
                return retValue;
        }

        /**
         *
         * @param positionOfTerm1
         * @param positionOfTerm2
         * @param winSize can exceed the size the the documents
         * @return
         */
        public static int unorderHALTimes(final int[] positionOfTerm1,
                        final int[] positionOfTerm2, final int winSize) {
                        int retValue =0;
                        if (positionOfTerm1 == null || positionOfTerm2 == null) {
                                return 0;
                        }
                        for(int i=0; i < positionOfTerm1.length; i++){
                                for(int j =0; j < positionOfTerm2.length; j++){
                                        int dist = winSize - (Math.abs(positionOfTerm1[i] -
                                            positionOfTerm2[j]) -1);
                                        if(dist > 0){
                                                retValue += dist;
                                        }
                                }
                        }
                        return retValue;
        }

        public static int unorderHALTimes(final int[] positionOfTerm1,
                        final int winSize) {
                        int retValue =0;
                        if (positionOfTerm1 == null || positionOfTerm1.length < 2) {
                                return 0;
                        }
                        for(int i=0; i < positionOfTerm1.length -1; i++){
                                for(int j =1; j < positionOfTerm1.length; j++){
                                        int dist = winSize - (Math.abs(positionOfTerm1[i] -
                                            positionOfTerm1[j]) -1);
                                        if(dist > 0){
                                                retValue += dist;
                                        }
                                }
                        }
                        return retValue;
        }


        /**
         * Counts number of blocks where two terms occur within a block of
         * windowSize in length, in a document of length documentLengthInTokens
         * where the blocks for the terms are as given
         *
         * @param blocksOfTerm1
         * @param start1
         *              The start index for the correct blockIds in blocksOfTerm1
         * @param end1
         *              The end for the correct blockIds in blocksOfTerm1
         * @param blocksOfTerm2
         * @param start2
         *              The start index for the correct blockIds in blocksOfTerm2
         * @param end2
         *              The end index for the correct blockIds in blocksOfTerm2
         * @param windowSize
         * @param documentLengthInTokens
         **/
        public static int noTimes(final int[] positionOfTerm1, int start1, int end1,
                        final int[] positionOfTerm2, int start2, int end2,
                        final int windowSize, final int documentLengthInTokens) {

                if (positionOfTerm1 == null) {
                        return 0;
                }

                if (positionOfTerm2 == null) {
                        return 0;
                }

                int numberOfNGrams = documentLengthInTokens < windowSize ? 1
                                : documentLengthInTokens - windowSize + 1;
                int count = 0;
                final int[] windows_for_term1 = new int[numberOfNGrams];
                final int[] windows_for_term2 = new int[numberOfNGrams];
                windowsForTerms(positionOfTerm1, start1, end1, windowSize,
                                numberOfNGrams, windows_for_term1);
                windowsForTerms(positionOfTerm2, start2, end2, windowSize,
                                numberOfNGrams, windows_for_term2);

                for (int i = 0; i < numberOfNGrams; i++) {
                        if (windows_for_term1[i] > 0 && windows_for_term2[i] > 0)
                                count++;
                }
                if(count > numberOfNGrams){
                        System.out.println("error:" +count + ",_" + numberOfNGrams);
                        System.exit(1);
                }
                return count;
        }

        public static int noTimes(final int[] positionOfTerm1,
                        final int[] positionOfTerm2,
                        final int windowSize, final int documentLengthInTokens) {
                return noTimes(positionOfTerm1, 0, positionOfTerm1.length,
                                positionOfTerm2, 0, positionOfTerm2.length,
                                windowSize, documentLengthInTokens);
        }

        public static int noTimes(final int[][] blocksOfTerms, int[] start,
                        int[] end, final int windowSize, final int documentLengthInTokens) {

                int numberOfTerms = blocksOfTerms.length;

                for (int i = 0; i < numberOfTerms; i++)
                        if (blocksOfTerms == null)
                                return 0;

                int numberOfNGrams = documentLengthInTokens < windowSize ? 1
                                : documentLengthInTokens - windowSize + 1;
                int count = 0;
                int[][] windows_for_terms = new int[numberOfTerms][numberOfNGrams];

                for (int i = 0; i < numberOfTerms; i++) {
                        windowsForTerms(blocksOfTerms[i], start[i], end[i], windowSize,
                                        numberOfNGrams, windows_for_terms[i]);
                }

                for (int i = 0; i < numberOfNGrams; i++) {
                        boolean flag = true;
                        for (int j = 0; j < numberOfTerms; j++)
                                if (!(windows_for_terms[j][i] > 0)) {
                                        flag = false;
                                        break;
                                }
                        if (flag)
                                count++;

                }
                return count;
        }

        /**
         * Count the bigram frequency given by a sigmoid function.
         *
         * @param blocksOfTerm1
         * @param start1
         * @param end1
         * @param blocksOfTerm2
         * @param start2
         * @param end2
         * @return
         */
        // public static double bigramFrequency(final int[] blocksOfTerm1, int
        // start1, int end1, final int[] blocksOfTerm2, int start2, int end2, int
        // wSize){
        // if( blocksOfTerm1 == null){
        // return 0;
        // }
```

```
//
// if(blocksOfTerm2 == null){
// return 0;
// }
// double nGf = 0d;
//
// if (end1-start1 >= end2-start2)
// for (int i=start1; i<end1; i++){
// nGf += bigramFrequency(blocksOfTerm1[i], blocksOfTerm2, start2, end2,
// wSize);
// }
// else
// for (int i=start2; i<end2; i++){
// nGf += bigramFrequency(blocksOfTerm2[i], blocksOfTerm1, start1, end1,
// wSize);
// }
//
// return nGf;
// }

// public static double bigramFrequency(int pos, final int[] blocksOfTerm,
// int start, int end, int wSize) {
// /*
// * int minDist = Statistics.sum(blocksOfTerm); for (int i=start; i<end;
// * i++){ minDist = Math.min(minDist, Math.abs(blocksOfTerm[i]-pos)); }
// * if (minDist<=0d) return 0d;
// */
//
// int minDist = Math.abs(blocksOfTerm[start] - pos);
// if (end - start == 1) {
// minDist = Math.abs(blocksOfTerm[start] - pos);
// } else if (end - start == 2)
// minDist = Math.min(Math.abs(blocksOfTerm[start] - pos), Math
// .abs(blocksOfTerm[end - 1] - pos));
//
// // if pos falls outside of the range of blocksOfTerm
// else if (pos < blocksOfTerm[start])
// minDist = Math.abs(blocksOfTerm[start] - pos);
// else if (pos > blocksOfTerm[end - 1])
// minDist = Math.abs(blocksOfTerm[end - 1] - pos);
// else {// perform a binary search
// // logger.debug("start binary search");
// int left = start;
// int right = end;
// int mid = (start + end - 1) / 2;
// while (true) {
// // logger.debug("start="+start+", mid="+mid+", end="+end);
// // logger.debug(blocksOfTerm[mid]+", "+pos+", "+blocksOfTerm[mid+1]);
// if (mid == left) {
// minDist = Math.min(Math.abs(blocksOfTerm[mid] - pos), Math
// .abs(blocksOfTerm[mid + 1] - pos));
// break;
// } else if (mid == right - 1) {
// minDist = Math.min(Math.abs(blocksOfTerm[mid] - pos), Math
// .abs(blocksOfTerm[mid - 1] - pos));
// break;
// } else if (pos > blocksOfTerm[mid]
// && pos < blocksOfTerm[mid + 1]) {
// minDist = Math.min(Math.abs(blocksOfTerm[mid] - pos), Math
// .abs(blocksOfTerm[mid + 1] - pos));
// break;
// } else if (pos == blocksOfTerm[mid]
// || pos == blocksOfTerm[mid + 1]) {
// return 0d;
// } else {
// if (pos < blocksOfTerm[mid]) {
// right = mid + 1;
// mid = (left + mid) / 2;
// } else {
// left = mid;
// mid = (mid + right - 1) / 2;
// }
// }
// }
// }
// if (wSize != 0 && (minDist <= 0 || minDist > wSize - 1))
// return 0d;
//
```

```
// return SigmoidFunction.inverseSigmoid((double) minDist, sigmoidPower);
// }

public static int getMinDist(int pos, final int[] blocksOfTerm, int start,
                int end, int wSize) {
        /*
         * int minDist = Statistics.sum(blocksOfTerm); for (int i=start; i<end;
         * i++){ minDist = Math.min(minDist, Math.abs(blocksOfTerm[i]-pos)); }
         * if (minDist<=0d) return 0d;
         */

        int minDist = Math.abs(blocksOfTerm[start] - pos);
        if (end - start == 1) {
                minDist = Math.abs(blocksOfTerm[start] - pos);
        } else if (end - start == 2)
                minDist = Math.min(Math.abs(blocksOfTerm[start] - pos), Math
                                .abs(blocksOfTerm[end - 1] - pos));

        // if pos falls outside of the range of blocksOfTerm
        else if (pos < blocksOfTerm[start])
                minDist = Math.abs(blocksOfTerm[start] - pos);
        else if (pos > blocksOfTerm[end - 1])
                minDist = Math.abs(blocksOfTerm[end - 1] - pos);
        else {// perform a binary search
                // logger.debug("start binary search");
                int left = start;
                int right = end;
                int mid = (start + end - 1) / 2;
                while (true) {
                        // logger.debug("start="+start+", mid="+mid+", end="+end);
                        // logger.debug(blocksOfTerm[mid]+", "+pos+", "+blocksOfTerm[mid
                             +1]);
                        if (mid == left) {
                                minDist = Math.min(Math.abs(blocksOfTerm[mid] - pos),
                                    Math
                                                .abs(blocksOfTerm[mid + 1] - pos));
                                break;
                        } else if (mid == right - 1) {
```

```
                                        minDist = Math.min(Math.abs(blocksOfTerm[mid] - pos),
                                              Math
                                                            .abs(blocksOfTerm[mid - 1] - pos));
                                        break;
                                } else if (pos > blocksOfTerm[mid]
                                                && pos < blocksOfTerm[mid + 1]) {
                                        minDist = Math.min(Math.abs(blocksOfTerm[mid] - pos),
                                              Math
                                                            .abs(blocksOfTerm[mid + 1] - pos));
                                        break;
                                } else if (pos == blocksOfTerm[mid]
                                                || pos == blocksOfTerm[mid + 1]) {
                                        return 0;
                                } else {
                                        if (pos < blocksOfTerm[mid]) {
                                                right = mid + 1;
                                                mid = (left + mid) / 2;
                                        } else {
                                                left = mid;
                                                mid = (mid + right - 1) / 2;
                                        }
                                }
                        }
                }

                if (wSize != 0 && (minDist <= 0 || minDist > wSize - 1))
                        return -1;

                return minDist - 1;
        }

        public static double bigramFrequency(final int[] positionOfTerm1,
                        final int[] positionOfTerm2,
                        final int windowSize) {
                return bigramFrequency(positionOfTerm1, 0, positionOfTerm1.length,
                                positionOfTerm2, 0, positionOfTerm2.length,
                                windowSize);
        }
        /**
```

```
 * Count the bigram frequency given by a sigmoid function.
 * @param blocksOfTerm1
 * @param start1
 * @param end1
 * @param blocksOfTerm2
 * @param start2
 * @param end2
 * @return
 */
public static double bigramFrequency(final int[] blocksOfTerm1, int start1, int end1,
    final int[] blocksOfTerm2, int start2, int end2, int wSize){
        if( blocksOfTerm1 == null){
                return 0;
        }

        if(blocksOfTerm2 == null){
                return 0;
        }
        double nGf = 0d;

        if (end1-start1 >= end2-start2)
                for (int i=start1; i<end1; i++){
                        nGf += bigramFrequency(blocksOfTerm1[i], blocksOfTerm2, start2,
                            end2, wSize);
                }
        else
                for (int i=start2; i<end2; i++){
                        nGf += bigramFrequency(blocksOfTerm2[i], blocksOfTerm1, start1,
                            end1, wSize);
                }

        return nGf;
}

public static double bigramFrequency(int pos, final int[] blocksOfTerm, int start, int
    end, int wSize){
        /*int minDist = Statistics.sum(blocksOfTerm);
        for (int i=start; i<end; i++){
                minDist = Math.min(minDist, Math.abs(blocksOfTerm[i]-pos));
```

```
}
if (minDist<=0d)
        return 0d;*/

int minDist = Math.abs(blocksOfTerm[start]-pos);
if (end-start==1){
        minDist = Math.abs(blocksOfTerm[start]-pos);
}
else if (end-start == 2)
        minDist = Math.min(Math.abs(blocksOfTerm[start]-pos), Math.abs(
            blocksOfTerm[end-1]-pos));

// if pos falls outside of the range of blocksOfTerm
else if (pos<blocksOfTerm[start])
        minDist = Math.abs(blocksOfTerm[start]-pos);
else if (pos>blocksOfTerm[end-1])
        minDist = Math.abs(blocksOfTerm[end-1]-pos);
else {// perform a binary search
        // logger.debug("start binary search");
        int left = start; int right = end;
        int mid = (start+end-1)/2;
        while (true){
                // logger.debug("start="+start+", mid="+mid+", end="+end);
                // logger.debug(blocksOfTerm[mid]+", "+pos+", "+blocksOfTerm[mid
                    +1]);
                if (mid==left){
                        minDist = Math.min(Math.abs(blocksOfTerm[mid]-pos), Math.
                            abs(blocksOfTerm[mid+1]-pos));
                        break;
                }else if (mid==right-1){
                        minDist = Math.min(Math.abs(blocksOfTerm[mid]-pos), Math.
                            abs(blocksOfTerm[mid-1]-pos));
                        break;
                }else if (pos>blocksOfTerm[mid] && pos<blocksOfTerm[mid+1]){
                        minDist = Math.min(Math.abs(blocksOfTerm[mid]-pos), Math.
                            abs(blocksOfTerm[mid+1]-pos));
                        break;
                }
                else if (pos == blocksOfTerm[mid] || pos == blocksOfTerm[mid+1]){
```

```
                                return 0d;
                        }
                        else{
                                if (pos<blocksOfTerm[mid]){
                                        right = mid+1;
                                        mid=(left+mid)/2;
                                }
                                else{
                                        left = mid;
                                        mid=(mid+right-1)/2;
                                }
                        }
                }
        }
        if ( wSize!=0 && (minDist<=0 || minDist > wSize-1))
                return 0d;

        return org.dutir.math.function.SigmoidFunction.inverseSigmoid((double)minDist,
             sigmoidPower);
}


public static void windowsForTerms(int[] blocksOfTerm, int start, int end,
                int windowSize, int numberOfNGrams, int[] windows_for_term) {

        // for each block
        for (int i = start; i < end; i++) {
                final int a = blocksOfTerm[i];
                int j;
                if (a - windowSize + 1 < 0)
                        j = 0;
                else
                        j = a - windowSize + 1;
                // for each window matching that block
                for (; j <= a && j < numberOfNGrams; j++) {
                        windows_for_term[j] = 1;
                }
        }
}
```

```
/** number of blocks where */
public static int noTimesSameOrder(final int[] blocksOfTerm1, int start1,
                int end1, final int[] blocksofTerm2, int start2, int end2,
                final int windowSize, final int documentLengthInTokens) {

        if (blocksOfTerm1 == null) {
                return 0;
        }

        if (blocksofTerm2 == null) {
                return 0;
        }

        final int numberOfNGrams = documentLengthInTokens < windowSize ? 1
                        : documentLengthInTokens - windowSize + 1;
        final boolean[] matchingWindows = new boolean[numberOfNGrams];

        for (int k1 = start1; k1 < end1; k1++) {
                for (int k2 = start2; k2 < end2; k2++) {
                        if (((blocksofTerm2[k2] - blocksOfTerm1[k1] < windowSize) && ((
                            blocksofTerm2[k2] - blocksOfTerm1[k1]) > 0))) {
                                final int len = blocksofTerm2.length;
                                for (int i = 0; i < len; i++) {
                                        final int a = blocksofTerm2[i];
                                        int j;
                                        if (a - windowSize + 1 < 0)
                                                j = 0;
                                        else
                                                j = a - windowSize + 1;
                                        // for each window matching that block
                                        for (; j <= a && j < numberOfNGrams; j++) {
                                                matchingWindows[j] = true;
                                        }
                                }
                        }
                }
        }
        int count = 0;

        for (int i = 0; i < documentLengthInTokens; i++) {
                if (matchingWindows[i])
                        count++;
        }
        return count;
}

public static int findSmallest(int[] x, int[] y) {
        Arrays.sort(x);
        Arrays.sort(y);

        int i = 0;
        int j = 0;
        int smallest = -1;

        while (true) {

                final int dif = Math.abs((x[i] - y[j]));

                if (smallest == -1) {
                        smallest = dif;
                } else if (dif < smallest) {
                        smallest = dif;
                } else if (dif == 0) {
                        return 0;
                }

                if (x[i] < y[j]) {
                        i++;
                        if (i == x.length)
                                return smallest;
                } else if (x[i] > y[j]) {
                        j++;
                        if (j == y.length)
                                return smallest;
                }
        }
        }

        public static void main(String args[]) {

//              int[] x = new int[] { 8, 14, 10, 15 };
//              int[] y = new int[] { 4, 6, 10, 12, 17, 1 };
//
//              System.out.println(findSmallest(x, y));

                double sd = Double.parseDouble(ApplicationSetup.getProperty("TermAssociation.sd",
                     "10"));
                NormalDistributionImpl nDist = new NormalDistributionImpl(0, sd);
                int[] pos1 = new int[]{1,7};
                int[] pos2 = new int[]{3};
                int winsize =1;
                int dl = 50;
//              System.out.println(Distance.noTimes(pos1, pos2, winsize, dl));
//              System.out.println(Distance.unorderHALTimes(pos1, pos2, winsize));
//              System.out.println(Distance.noTimes(pos1, pos2, winsize, dl));
                System.out.println(Distance.unorderGaussianTimes(pos1, pos2, winsize, nDist));
        }
}
```

# Appendix E

## Related Java code for the TopPRF work

Again, cecause different researchers use different IR systems for their work, so here we just show some related code for different usage of topic information in PRF. In this thesis, we use method 10-12 for the work in Chapter 4.

```
/**
 *
 */
package org.apache.lucene.postProcess.termselector;

import gnu.trove.TLongObjectHashMap;
import gnu.trove.TObjectIntHashMap;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Random;
import java.util.Set;

import org.apache.commons.math.stat.descriptive.moment.StandardDeviation;
import org.apache.log4j.Logger;
import org.apache.lucene.MetricsUtils;
import org.apache.lucene.index.IndexReader;
import org.apache.lucene.index.Term;
import org.apache.lucene.index.TermDocs;
import org.apache.lucene.index.TermFreqVector;
import org.apache.lucene.index.TermPositionVector;
import org.apache.lucene.postProcess.QueryExpansionModel;
import org.apache.lucene.postProcess.termselector.LatentDirichletAllocation.GibbsSample;
import org.apache.lucene.search.model.Idf;
import org.dutir.lucene.util.ApplicationSetup;
import org.dutir.lucene.util.ExpansionTerms.ExpansionTerm;
import org.dutir.lucene.util.Rounding;
import org.dutir.lucene.util.TermsCache;
import org.dutir.util.Arrays;
import org.dutir.util.Normalizer;
import org.dutir.util.symbol.MapSymbolTable;
import org.dutir.util.symbol.SymbolTable;




/**
 * @author Jun Miao
 *
 */

public class TopicBasedTermSelector extends TermSelector {
        private static Logger logger = Logger.getLogger(TopicBasedTermSelector.class);
        static boolean LanguageModel = Boolean.parseBoolean(ApplicationSetup
                        .getProperty("Lucene.Search.LanguageModel", "false"));
        static int strategy = Integer.parseInt(ApplicationSetup.getProperty(
                        "TopicBasedTermSelector.strategy", "3"));
        static Boolean dataFromFile = Boolean.parseBoolean(ApplicationSetup.getProperty(
            "TopicBasedTermSelector.dataFromFile", "false"));

        static boolean expTag = Boolean.parseBoolean(ApplicationSetup.getProperty(
                        "TopicBasedTermSelector.expTag", "false"));
        static boolean withOrgScore = Boolean.parseBoolean(ApplicationSetup.getProperty(
                    "TopicBasedTermSelector.withOrgScore", "false"));
        static String topicDataPath = ApplicationSetup.getProperty(
            "TopicBasedTermSelector.topicDataPath", "docTopics.txt");
```

```
static int expNum = Integer.parseInt(ApplicationSetup.getProperty(
            "TopicBasedTermSelector.expNum", "15"));
static int expDoc = Integer.parseInt(ApplicationSetup.getProperty(
            "TopicBasedTermSelector.expDoc", "10"));

static boolean associationTag = Boolean.parseBoolean(ApplicationSetup
            .getProperty("TopicBasedTermSelector.associationTag", "false"));

static float threshold = Float.parseFloat(ApplicationSetup.getProperty(
            "TopicBasedTermSelector.threshold", "0.2"));
static float lambda = Float.parseFloat(ApplicationSetup.getProperty(
            "TopicBasedTermSelector.lambda", "0.5"));
static float beta = Float.parseFloat(ApplicationSetup.getProperty(
            "TopicBasedTermSelector.beta", "0.3"));
static int winSize = Integer.parseInt(ApplicationSetup.getProperty(
            "Association.winSize", "50"));

static Random RANDOM = new Random(43);
static short NUM_TOPICS = Short.parseShort(ApplicationSetup.getProperty(
            "TopicBasedTermSelector.NUM_TOPICS", "5"));

static String word2vecDataPath = ApplicationSetup.getProperty(
            "TopicBasedTermSelector.word2vecDataPath", "text8.model.txt");

// static double DOC_TOPIC_PRIOR = 0.01;
static double TOPIC_WORD_PRIOR = 0.01;
static double DOC_TOPIC_PRIOR = 2d / NUM_TOPICS;
static int numSamples = 30;
static int burnin = 30;
static int sampleLag = 10;

static int BURNIN_EPOCHS = 10;
static int SAMPLE_LAG = 30;
static int NUM_SAMPLES = 30;
//static HashMap<String, float[]> vectorOfTerms = new HashMap<String, float[]>();
static HashMap<String, float[]> vectorOfTerms = null;
protected int EXPANSION_MIN_DOCUMENTS;
float dscores[];
```

```
TObjectIntHashMap<String> dfMap = null;

public TopicBasedTermSelector() throws IOException {
        super();
        this.setMetaInfo("normalize.weights", "true");
        this.EXPANSION_MIN_DOCUMENTS = Integer.parseInt(ApplicationSetup
                        .getProperty("expansion.mindocuments", "2"));
        //read training data of word2vec
        String vectline;
        if (vectorOfTerms != null){
                br = new BufferedReader(new FileReader(word2vecDataPath));
                vectline = br.readLine();
                int vctDimension = Integer.parseInt(vectline.split("\\s+")[1]);

                while  ((vectline = br.readLine()) != null){
                        //logger.info("Start reading word2vec file   " + this.
                             word2vecDataPath);
                        String [] pairs = vectline.split("\\s+");
                        String term = pairs[0];
                        float [] vector = new float[vctDimension];

                        for (int k = 0; k < vctDimension; k++)
                                vector[k] = Float.parseFloat(pairs[k+1]);

                        TopicBasedTermSelector.vectorOfTerms.put(term, vector);

                }//end of read word2vec file
        }
}

/*
 * (non-Javadoc)
 *
 * @see
 * org.apache.lucene.postProcess.termselector.TermSelector#assignTermWeights
 * (int[], org.apache.lucene.postProcess.QueryExpansionModel)
 */
@Override
```

```
public void assignTermWeights(int[] docids, float scores[],
                QueryExpansionModel QEModel) throws IOException {
        dscores = new float[scores.length];
        System.arraycopy(scores, 0, dscores, 0, scores.length);

        if (LanguageModel) {
                indriNorm(dscores);
        }
        Normalizer.norm2(dscores);
logger.info("sum_of_doc_weights:" + Arrays.sum(dscores));
Normalizer.norm_MaxMin_0_1(dscores); //normalized doc scores
if (logger.isInfoEnabled())
{
    StringBuffer buf = new StringBuffer();
    for (int i = 0; i < dscores.length; i++) {
        buf.append("" + dscores[i] + ",_");
    }
    logger.info("4.doc_weights:" + buf.toString());
}
        String[][] termCache = null;
        int[][] termFreq = null;
        termMap = new HashMap<String, ExpansionTerm>();
        this.feedbackSetLength = 0;
        termCache = new String[docids.length][];
        termFreq = new int[docids.length][];
        dfMap = new TObjectIntHashMap<String>();
        for (int i = 0; i < docids.length; i++) {
                int docid = docids[i];
                System.out.println(docids[i]);
                TermFreqVector tfv = null;
                try {
                        tfv = this.searcher.getIndexReader().getTermFreqVector(docid,
                                        field);
                } catch (IOException e) {
                        e.printStackTrace();
                }
                if (tfv == null)
                        logger.warn("document_" + docid + "_not_found,_field=" + field);
                else {
```

```
                String strterms[] = tfv.getTerms();
                int freqs[] = tfv.getTermFrequencies();
                termCache[i] = strterms;
                termFreq[i] = freqs;
        }
}


// //////////LDA clustering/////////////////////
MapSymbolTable SYMBOL_TABLE = new MapSymbolTable();
int[][] DOC_WORDS = new int[docids.length][];
int querytermid[] = new int[this.originalQueryTermidSet.size()];
int pos = 0;
int backids[] = new int[0];
if (expTag) {
        int reallExp = Math.min(expDoc, docids.length);
        int expDocs[] = new int[reallExp];
        float expscores[] = new float[reallExp];
        System.arraycopy(docids, 0, expDocs, 0, reallExp);
        System.arraycopy(scores, 0, expscores, 0, reallExp);
        TermSelector selector = TermSelector.getTermSelector(
                        "RocchioTermSelector", this.searcher);
        selector.setField(field);
        selector.setMetaInfo("normalize.weights", "false");
        selector.setOriginalQueryTerms(originalQueryTermidSet);
        selector.assignTermWeights(expDocs, expscores, QEModel);

        HashMap<String, ExpansionTerm> map = selector
                        .getMostWeightedTermsInHashMap(expNum);
        assert map.size() <= expNum;
        Set<String> keyset = new HashSet<String>(map.keySet());
        keyset.addAll(this.originalQueryTermidSet);
        querytermid = new int[keyset.size()];
        for (String term : keyset) {
                int id = SYMBOL_TABLE.getOrAddSymbol(term);
                querytermid[pos++] = id;
        }
```

```
} else {
        for (String term : this.originalQueryTermidSet) {
                int id = SYMBOL_TABLE.getOrAddSymbol(term);
                querytermid[pos++] = id;
        }
}

for (int i = 0; i < docids.length; i++) {
        int len = Arrays.sum(termFreq[i]);
        DOC_WORDS[i] = new int[len];
        pos = 0;
        for (int j = 0; j < termCache[i].length; j++) {
                String term = termCache[i][j];
                dfMap.adjustOrPutValue(term, 1, 1);
                int id = SYMBOL_TABLE.getOrAddSymbol(term);
                for (int k = 0; k < termFreq[i][j]; k++) {
                        DOC_WORDS[i][pos++] = id;
                }
        }
        assert len == pos;
}
for (int[] words : DOC_WORDS)
        Arrays.permute(words, RANDOM);
// LdaReportingHandler handler = new LdaReportingHandler(SYMBOL_TABLE);

// get a co-occurrence lookup map. ////////////
MapSymbolTable coTable = SYMBOL_TABLE.clone();
TermAssociation tAss = null;
if (associationTag) {
        tAss = TermAssociation.built(this.searcher, this.topDoc, coTable,
                        this.field, winSize);
}

// ////////////////////////////////////////////

// LatentDirichletAllocation.GibbsSample sample =
// LatentDirichletAllocation
// .gibbsSampler(DOC_WORDS, NUM_TOPICS, DOC_TOPIC_PRIOR,
// TOPIC_WORD_PRIOR, BURNIN_EPOCHS, SAMPLE_LAG,
            // NUM_SAMPLES, RANDOM, querytermid, backids, null, tAss);
                        LatentDirichletAllocation.GibbsSample sample =
                               LatentDirichletAllocation
                        .gibbsSampler(DOC_WORDS, NUM_TOPICS, DOC_TOPIC_PRIOR,
                                        TOPIC_WORD_PRIOR, BURNIN_EPOCHS, SAMPLE_LAG,
                                        NUM_SAMPLES, RANDOM, querytermid, backids, null);

            LatentDirichletAllocation lda = sample.lda();
                        short[][] qsamples = lda.sampleTopics(querytermid, numSamples,
                               burnin,
                        sampleLag, RANDOM);
            float theta[] = new float[NUM_TOPICS];
            java.util.Arrays.fill(theta, 0);
            for (int i = 0; i < qsamples.length; i++) {
                    for (int j = 0; j < qsamples[i].length; j++) {
                            theta[qsamples[i][j]]++;
                    }
            }
            float total = querytermid.length * numSamples;
            for (int i = 0; i < theta.length; i++) {
                    theta[i] = theta[i] / total;
            }

            selectTerm(SYMBOL_TABLE, sample, QEModel,
                        theta, querytermid, lda, tAss, docids);
    }

    float[] sampleTheta(int numTopics, LatentDirichletAllocation lda,
                    int[] words) {
            short[][] qsamples = lda.sampleTopics(words, numSamples, burnin,
                        sampleLag, RANDOM);

            float theta[] = new float[numTopics];
            java.util.Arrays.fill(theta, 0);

            for (int i = 0; i < qsamples.length; i++) {
                    for (int j = 0; j < qsamples[i].length; j++) {
                            theta[qsamples[i][j]]++;
                    }
```

```
        }
        float total = words.length * numSamples;
        for (int i = 0; i < theta.length; i++) {
                theta[i] = theta[i] / total;
        }
        return theta;
}

float[][] sampleThetas(int times, int numTopics,
                LatentDirichletAllocation lda, int[] words) {
        float retValue[][] = new float[times][];
        for (int i = 0; i < times; i++) {
                retValue[i] = sampleTheta(numTopics, lda, words);
        }
        return retValue;
}

float[] sampleThetasAver(int times, int numTopics,
                LatentDirichletAllocation lda, int[] words) {
        float[][] aver = sampleThetas(times, numTopics, lda, words);
        float retV[] = new float[numTopics];
        for (int i = 0; i < aver.length; i++) {
                for (int j = 0; j < aver[0].length; j++) {
                        retV[j] = aver[i][j];
                }
        }
        for (int i = 0; i < numTopics; i++) {
                retV[i] = retV[i] / numTopics;
        }
        return retV;
}

private ExpansionTerm[] selectTerm(SymbolTable SYMBOL_TABLE,
                GibbsSample sample, QueryExpansionModel QEModel, float theta[],
                int querytermid[], LatentDirichletAllocation lda,
                TermAssociation tAss,
                int []docIds) throws IOException {

        ExpansionTerm[] allTerms = new ExpansionTerm[SYMBOL_TABLE.numSymbols()];
```

```
int index[] = Arrays.indexSort(theta);

double docTopicProbs[][] = null;
      if (dataFromFile){
          //TODO:read doc topic matrix from the file to a hashmap

          TLongObjectHashMap<double[]> docTopics = new TLongObjectHashMap<double[]>();
          File topicData = new File(topicDataPath);
          if (!topicData.exists())
               throw new IOException("Doc_Topic_data_can't_be_found_from_path_" +
                    topicDataPath);

          logger.info("Start_loading_Doc/Topic_info_from_" + topicDataPath);
      BufferedReader br = new BufferedReader(new FileReader(topicData));
      String eachDoc = null;
      int topicNum = 0;
      while ((eachDoc = br.readLine()) != null) {
          String [] data = eachDoc.split("\t");
          int docId = Integer.parseInt(data[0]);
          double[] topicProbs = new double[data.length - 1];
          for (int i = 1; i < data.length; i++)
              topicProbs[i - 1] = Double.parseDouble(data[i].trim());

          docTopics.put(docId, topicProbs);
          if (topicNum == 0)
              topicNum = topicProbs.length;
       }
      br.close();
      logger.info("Loading_completed");
      docTopicProbs = new double[docIds.length][topicNum];
      //Get topic probs for each docs
      logger.info("Getting_topic_vectors_for_feedback_docs");
          for (int i = 0; i< docIds.length; i++){
              if (docTopics.get(docIds[i]) != null){
                  docTopicProbs[i] = docTopics.get(docIds[i]);
                  logger.debug("topic_vector_lenght_for_doc" + docIds[i] + "is_" +
                       docTopicProbs[i].length );
              }
              else
```

```
                throw new IllegalStateException("Can't␣find␣topic␣probabilities␣for␣doc
                    ␣" + docIds[i]);
            }
            logger.info("Getting␣topic␣vectors␣for␣feedback␣docs␣is␣done!");

}else{
    docTopicProbs = new double[sample.numDocuments()][sample.numTopics()];

    for (int i = 0; i < index.length; i++) {
        float prob = 0;
        for (int j = 0; j < querytermid.length; j++) {
            prob += Idf.log(sample.topicWordProb(index[i],
                    querytermid[j]));
        }
        if (logger.isDebugEnabled())
            logger.debug("topic:␣" + index[i] + "␣-␣" + theta[index[i]]
                    + ",␣topicCount:␣" + sample.topicCount(index[i])
                    + ",␣prob:␣" + prob);
    }
    StringBuilder buf = new StringBuilder();

    for (int i = 0, n = sample.numDocuments(); i < n; i++) {
        buf.append(i + ":\t");
        for (int j = 0; j < sample.numTopics(); j++) {
            buf.append(Rounding.round(sample.documentTopicProb(i, j), 5)
                    + "\t");
            docTopicProbs[i][j] =  Rounding.round(sample.documentTopicProb(i, j), 5);
        }
        buf.append("\n");
    }
    if (logger.isDebugEnabled())
        logger.debug("doc␣topic␣distribution\n" + buf.toString());

}



      final int len = allTerms.length;
      int maxTopic = index[index.length - 1];
      if (logger.isDebugEnabled())
```

```
                logger.debug("max_topic: " + maxTopic);



        Iterator<String> it = originalQueryTermidSet.iterator();
String[] qterms = new String[originalQueryTermidSet.size()];
int qCount = 0;
while (it.hasNext()) {
        qterms[qCount] = (String) it.next();
        qCount++;
}





if (strategy == 1) { // take advantage of the topic with the highest
                                                        // prob
                float totalweight = 0;
                int feedbackNum = sample.numDocuments();
                for (int i = 0; i < len; i++) {
                        String term = SYMBOL_TABLE.idToSymbol(i);
                        TermsCache.Item item = getItem(term);
                        float TF = item.ctf;
                        float DF = item.df;
                        float weight = 0;

                        weight = 0;
                        // use probability in top 1 topic as original weight in one
                        // feedback doc, add all the score up and divide by the doc num
                        // then rank
                        for (int d = 0; d < feedbackNum; d++) {
                                double docProb = sample.docWordCount(d, i)
                                                / (float) sample.documentLength(d);
                                if (docProb == 0) {
                                        continue;
                                }
                                double topicProb = sample.topicWordProb(maxTopic, i);
                                double onedocWeight = (1 - beta) * docProb + beta
                                                * topicProb;
                                System.out.println("topic_weight_is_" + topicProb +
                                        "and_doc_weight_" + onedocWeight);
                                // one doc weight is the original doc weight smoothed by
                                     the
                                // topic prob
                                QEModel.setTotalDocumentLength(1);
                                weight += QEModel.score((float) onedocWeight, TF, DF);
                        }
                        weight /= feedbackNum;
                        if (dfMap.get(term) < EXPANSION_MIN_DOCUMENTS) {
                                weight = 0;
                        }
                        allTerms[i] = new ExpansionTerm(term, 0);
                        allTerms[i].setWeightExpansion(weight);
                        this.termMap.put(term, allTerms[i]);
                        totalweight += weight;
                }

                java.util.Arrays.sort(allTerms);
                // determine double normalizing factor
                float normaliser = allTerms[0].getWeightExpansion();
                for (ExpansionTerm term : allTerms) {
                        if (normaliser != 0) {
                                term.setWeightExpansion(term.getWeightExpansion()
                                                / totalweight);
                        }
                }
        } else if (strategy == 2) { // add by Jun Miao
                // take advantage of the topic with the highest
                // prob and rank terms by their deviation in topics
                float totalweight = 0;
                int feedbackNum = sample.numDocuments();
                for (int i = 0; i < len; i++) {
                        String term = SYMBOL_TABLE.idToSymbol(i);
                        TermsCache.Item item = getItem(term);
                        float TF = item.ctf;
                        float DF = item.df;
                        float weight = 0;

                        weight = 0;
                double []topicProb = new double[index.length];

                StandardDeviation sd =  new StandardDeviation();
                double termDeviation = sd.evaluate(topicProb);

                //double topicWeight = (float) Math.sqrt(termDeviation * sample.
                    topicWordProb(maxTopic, i));
                double topicWeight = (float)(termDeviation);
//              System.out.println("topic weight is " + sample.topicWordProb(
    maxTopic, i) +
//                      " and term deviation is " + termDeviation);

                for (int d = 0; d < feedbackNum; d++) {
                        double docProb = sample.docWordCount(d, i)
                                        / (float) sample.documentLength(d);
                        if (docProb == 0) {
                                continue;
                        }
                        double onedocWeight = (1 - beta) * docProb + beta
                                        * topicWeight;
//              System.out.println("topic weight is " + topicWeight +
//                  "and doc weight " + onedocWeight);
                        // one doc weight is the original doc weight smoothed by
                            the
                        // topic prob
                        QEModel.setTotalDocumentLength(1);
                        weight += QEModel.score((float) onedocWeight, TF, DF);
                }
                weight /= feedbackNum;

                if (dfMap.get(term) < EXPANSION_MIN_DOCUMENTS) {
                        weight = 0;
                }
                allTerms[i] = new ExpansionTerm(term, 0);
                allTerms[i].setWeightExpansion(weight);
                this.termMap.put(term, allTerms[i]);
                totalweight += weight;
        }
        java.util.Arrays.sort(allTerms);
        // determine double normalizing factor
        float normaliser = allTerms[0].getWeightExpansion();
        for (ExpansionTerm term : allTerms) {
                if (normaliser != 0) {
                        term.setWeightExpansion(term.getWeightExpansion()
                                        / totalweight);
                }
        }

} else if (strategy == 3) { // add by Jun Miao
        // take advantage of the topic with the highest
        // prob and rank terms by PMI in the collection

        try {
                float totalweight = 0;
                IndexReader ir = this.searcher.getIndexReader();

int docInColl = ir.numDocs();
                // get query collection probability P(q)
double[] qCollProb = new double[qterms.length];

// get the postingList and of query terms
ArrayList<HashSet<Integer>> qTermPostings = new ArrayList<HashSet<Integer>>();
for (qCount = 0; qCount < qterms.length; qCount++) {
    Term qterm = new Term(this.field, qterms[qCount]);
    TermDocs queryPostingIds;
    queryPostingIds = ir.termDocs(qterm);

    HashSet<Integer> qPostings = new HashSet<Integer>();

    while (queryPostingIds.next())
        qPostings.add(queryPostingIds.doc());

    qTermPostings.add(qPostings);
    qCollProb[qCount] = qPostings.size()
            / (double) docInColl;
}
```

```
for (int i = 0; i < len; i++) {
        String term = SYMBOL_TABLE.idToSymbol(i);
        float weight = 0;



        // Get the posting list of the current term
        weight = 0;


        Term currTerm = new Term(this.field, term);
        TermDocs currTermDocIds = ir.termDocs(currTerm);
        int currTermPostingNum = 0;
        int[] commonPostingNum = new int[qterms.length];


        while (currTermDocIds.next()) {
                for (int l = 0; l < qterms.length; l++) {
                        int id = currTermDocIds.doc();
                        if (qTermPostings.get(l).contains(id))
                                commonPostingNum[l]++;
                }
                currTermPostingNum++;
        }
        // Get all the P(term), P(term, qterm)
        // Calculate the PMI^2 weight of this term
        double currTermCollProb = (double) currTermPostingNum
                        / docInColl;
        double[] jointProbs = new double[qterms.length];
        for (int l = 0; l < qterms.length; l++) {
                jointProbs[l] = (double) commonPostingNum[l]
                                / docInColl;
                weight += (Math.log(jointProbs[l]
                                / (qCollProb[l] *
                                    currTermCollProb)))
                                / (-jointProbs[l]);
        }


        double topicWeight = sample.topicWordProb(maxTopic, i);
        weight /= qterms.length * topicWeight;
```

```
                            if (dfMap.get(term) < EXPANSION_MIN_DOCUMENTS) {
                                    weight = 0;
                            }
                            allTerms[i] = new ExpansionTerm(term, 0);
                            allTerms[i].setWeightExpansion(weight);
                            this.termMap.put(term, allTerms[i]);
                            totalweight += weight;
                    }

                    java.util.Arrays.sort(allTerms);
                    // determine double normalizing factor
                    float normaliser = allTerms[0].getWeightExpansion();
                    for (ExpansionTerm term : allTerms) {
                            if (normaliser != 0) {
                                    term.setWeightExpansion(term.getWeightExpansion()
                                                    / totalweight);
                            }
                    }

            } catch (IOException e) {
                    e.printStackTrace();
            }

    } else if (strategy == 4) { // add by Jun Miao
            // take advantage of word2Vec

            float totalweight = 0;
for (int i = 0; i < len; i++) {
    String term = SYMBOL_TABLE.idToSymbol(i);
    float weight = 0;

    //calculate the weight of a feedback term based on it's word2vec
    //similarity to query terms
    float [] termVector = TopicBasedTermSelector.vectorOfTerms.get(term);
    for (int t = 0; t < qterms.length; t++){
            float [] qtermVector = TopicBasedTermSelector.vectorOfTerms.get(qterms[t
                ]);
            weight += cosine_similarity(termVector, qtermVector);
    }
            weight /= qterms.length;

    if (dfMap.get(term) < EXPANSION_MIN_DOCUMENTS) {
            weight = 0;
    }
    allTerms[i] = new ExpansionTerm(term, 0);
    allTerms[i].setWeightExpansion(weight);
    this.termMap.put(term, allTerms[i]);
    totalweight += weight;
}

java.util.Arrays.sort(allTerms);
// determine double normalizing factor
float normaliser = allTerms[0].getWeightExpansion();
for (ExpansionTerm term : allTerms) {
    if (normaliser != 0) {
            term.setWeightExpansion(term.getWeightExpansion()
                            / totalweight);
    }
}

    }else if (strategy == 5) { // add by Jun Miao
            // compare the probability distribution of query terms and
            //candidate feedback terms.

        // get query terms topic probabilities

     float qtermTopicProb[][] = new float[qterms.length][index.length];
     for(int m = 0; m < qterms.length; m++){
         int id = SYMBOL_TABLE.symbolToID(qterms[m]);
         for (int k = 0; k < index.length; k++)
             qtermTopicProb[m][k] = (float) sample.topicWordProb(index[k], id);
     }

        float totalweight = 0;


int feedbackNum = sample.numDocuments();
```

```
        for (int i = 0; i < len; i++) {
            String term = SYMBOL_TABLE.idToSymbol(i);
            TermsCache.Item item = getItem(term);
            float TF = item.ctf;
            float DF = item.df;
            float weight = 0;
            double topicWeight = 0;



            //get the topic distribution of current term
            float []topicProb = new float[index.length];
            for (int k = 0; k < index.length; k++)
                topicProb[k] = (float) sample.topicWordProb(index[k], i);

            for(int m = 0; m < qterms.length; m++){
              topicWeight += cosine_similarity(topicProb, qtermTopicProb[m]);
            }

            topicWeight /= qterms.length;


            for (int d = 0; d < feedbackNum; d++) {
              double docProb = sample.docWordCount(d, i)
                        / (float) sample.documentLength(d);
              if (docProb == 0) {
                    continue;
              }
              double docWeight = QEModel.score((float) docProb, TF, DF);

              //topicWeight and doc weight are not on the same level
//                double onedocWeight = (1 - beta) * docWeight + beta
//                             * topicWeight;

             // double onedocWeight = (1 + beta * topicWeight) * docWeight;
              double onedocWeight = beta * (1 +  topicWeight)* sample.topicWordProb(
                  maxTopic, i)
                    + (1 - beta) * docWeight;
              // one doc weight is the original doc weight smoothed by the
            // topic prob
            QEModel.setTotalDocumentLength(1);
            //weight += QEModel.score((float) onedocWeight, TF, DF);
            weight += onedocWeight;
        }
        weight /= feedbackNum;

            if (dfMap.get(term) < EXPANSION_MIN_DOCUMENTS) {
                    weight = 0;
            }
            allTerms[i] = new ExpansionTerm(term, 0);
            allTerms[i].setWeightExpansion(weight);
            this.termMap.put(term, allTerms[i]);
            totalweight += weight;
    }
    java.util.Arrays.sort(allTerms);
    // determine double normalizing factor
    float normaliser = allTerms[0].getWeightExpansion();
    for (ExpansionTerm term : allTerms) {
        if (normaliser != 0) {
                term.setWeightExpansion(term.getWeightExpansion()
                                / totalweight);
        }
    }

        }else if (strategy >9){
    // term weight based on feedback quality
            System.out.println("I_am_in_strategy_" + strategy + "\n");

        double []feedDocScores = getFeedDocScores(strategy, sample.numDocuments(),
                dscores, docTopicProbs, withOrgScore, beta);

        for (int i = 0; i < feedDocScores.length; i++)
            System.out.println("the_score_of_feedback_doc_" + i + "_is_" + feedDocScores[
                i] + "\n");

    float totalweight = 0;
    int feedbackNum = sample.numDocuments();
    for (int i = 0; i < len; i++) {
        String term = SYMBOL_TABLE.idToSymbol(i);
        TermsCache.Item item = getItem(term);
        float TF = item.ctf;
        float DF = item.df;
        float weight = 0;

        weight = 0;
        // use probability in top 1 topic as original weight in one
        // feedback doc, add all the score up and divide by the doc num
        // then rank
        for (int d = 0; d < feedbackNum; d++) {
            double docProb = sample.docWordCount(d, i)
                    / (float) sample.documentLength(d);
            if (docProb == 0) {
                continue;
            }
            double onedocWeight = docProb * feedDocScores[d]; //Multiple the term prob by
                 the doc score

            QEModel.setTotalDocumentLength(1);
            weight += QEModel.score((float) onedocWeight, TF, DF);
        }
        weight /= feedbackNum;
        if (dfMap.get(term) < EXPANSION_MIN_DOCUMENTS) {
            weight = 0;
        }
        allTerms[i] = new ExpansionTerm(term, 0);
        allTerms[i].setWeightExpansion(weight);
        this.termMap.put(term, allTerms[i]);
        totalweight += weight;
    }

    java.util.Arrays.sort(allTerms);
    // determine double normalizing factor
    float normaliser = allTerms[0].getWeightExpansion();
    for (ExpansionTerm term : allTerms) {
        if (normaliser != 0) {
            term.setWeightExpansion(term.getWeightExpansion()
                    / totalweight);
                }
            }




                }
                return allTerms;
        }

        static String dmu = ApplicationSetup.getProperty("dlm.mu", "500");
        static float mu = Integer.parseInt(ApplicationSetup.getProperty(
                        "topicSL.mu", dmu));
private BufferedReader br;
// float numOfTokens = this.searcher.getNumTokens(field);
        public float score(float tf, float docLength, float termFrequency,
                        float numberOfTokens) {
                float pc = termFrequency / numberOfTokens;
                return (tf + mu * pc) / (docLength + mu);
        }

        private void indriNorm(float[] pQ) {

                float K = pQ[0]; // first is max
                float sum = 0;
                for (int i = 0; i < pQ.length; i++) {
                        pQ[i] = Idf.exp(K + pQ[i]);
                        sum += pQ[i];
                }
                for (int i = 0; i < pQ.length; i++) {
                        pQ[i] /= sum;
                }
        }

        /*
         * (non-Javadoc)
         *
         * @see org.apache.lucene.postProcess.termselector.TermSelector#getInfo()
         */
        @Override
```

```
public String getInfo() {
        return "TopicSel_s=" + strategy + "t=" + NUM_TOPICS + "beta=" + beta
                        + "expTag=" + expTag;
}

private double[] getFeedDocScores(int strategy, int feedDocNum, float[] orgDocScores,
     double[][] docTopicProbs,
        boolean withOrgScore, float docScoreAlpha){
    if (orgDocScores.length == 0)
        return null;
    double[] feedbackDocScores = new double[feedDocNum];
    java.util.Arrays.fill(feedbackDocScores, 1);
    int topicNum = docTopicProbs[0].length;

    if ((feedDocNum <= 3) || (strategy > 13) || (strategy < 10))
        return feedbackDocScores;

    switch(strategy){
        case 10://doc similarity score
            for (int i = 3; i< feedDocNum; i++){
                for(int j = 0; j < 3; j++)
                    feedbackDocScores[i] += getCosineSimilarity(docTopicProbs[i],
                        docTopicProbs[j]);

                feedbackDocScores[i] = (1 + (feedbackDocScores[i] - 1)/3)/2; //normalized
                     to range 0-1
            }
            break;
        case 11: //Norm2
            for (int i = 3; i< feedDocNum; i++){
            for(int j = 0; j < 3; j++)
                feedbackDocScores[i] += MetricsUtils.distL2(docTopicProbs[i],
                    docTopicProbs[j]);

            feedbackDocScores[i] = feedbackDocScores[i] /(3*topicNum); //normalized to
                 range 0-1
        }
        break;
        case 12:// topic entropy. Smaller, better
```

```
            for (int i = 0; i< feedDocNum; i++){
                    for(int j = 0; j< topicNum; j++)
                feedbackDocScores[i] += -1 * docTopicProbs[i][j] * Math.log(docTopicProbs
                    [i][j]);

            for (int k = 0; k< feedDocNum; k++)
                feedbackDocScores[k] = 1- (feedbackDocScores[k]/Math.log(topicNum));
        }
        break;
        case 13: //Jenson-Shannon divergence from avg distribution
            double[] avgTopicDist = new double[topicNum];
            for (int i = 0; i< feedDocNum; i++)
                for(int j = 0; j< topicNum; j++)
                    avgTopicDist[j] += docTopicProbs[i][j];

            for (int i = 0; i< feedDocNum; i++)
                avgTopicDist[i] /= feedDocNum;

            for (int i = 0; i< feedDocNum; i++)
            feedbackDocScores[i] = MetricsUtils.jsd(docTopicProbs[i], avgTopicDist);

        break;

     default: break;

    }

    if (withOrgScore)
        for (int i = 0; i< feedDocNum; i++)
            feedbackDocScores[i] = feedbackDocScores[i] * (1 - docScoreAlpha)
            + docScoreAlpha * orgDocScores[i];

    return feedbackDocScores;
}

private static double cosine_similarity(float[] vec1, float[] vec2) {
double dp = dot_product(vec1,vec2);
double magnitudeA = find_magnitude(vec1);
double magnitudeB = find_magnitude(vec2);
```

```
    return (dp)/(magnitudeA*magnitudeB);
}

        private static double getCosineSimilarity(double[] vec1, double[] vec2) {
             double dp = dot_product(vec1,vec2);
             double magnitudeA = find_magnitude(vec1);
             double magnitudeB = find_magnitude(vec2);
             return (dp)/(magnitudeA*magnitudeB);
         }

private static double find_magnitude(float[] vec) {
    double sum_mag=0;
    for(int i=0;i<vec.length;i++)
    {
        sum_mag = sum_mag + vec[i]*vec[i];
    }
    return Math.sqrt(sum_mag);
}

private static double find_magnitude(double[] vec) {
    double sum_mag=0;
    for(int i=0;i<vec.length;i++)
    {
        sum_mag = sum_mag + vec[i]*vec[i];
    }
    return Math.sqrt(sum_mag);
}

private static double dot_product(float[] vec1, float[] vec2) {
    double sum=0;
    for(int i=0;i<vec1.length;i++)
    {
        sum = sum + vec1[i]*vec2[i];
    }
    return sum;
}

private static double dot_product(double[] vec1, double[] vec2) {
    double sum=0;
```

```
        for(int i=0;i<vec1.length;i++)
        {
            sum = sum + vec1[i]*vec2[i];
        }
        return sum;
    }

        /**
         * @param args
         */
        public static void main(String[] args) {

        }

        @Override
        public void assignTermWeights(String[][] terms, int[][] freqs,
                        TermPositionVector[] tfvs, QueryExpansionModel QEModel) {
                // TODO Auto-generated method stub

        }

}
```