TASK ORIENTED TOOLS FOR INFORMATION RETRIEVAL

by

Peilin Yang

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering

Summer 2017

TASK ORIENTED TOOLS FOR INFORMATION RETRIEVAL

by

Peilin Yang

Approved: _____
Kenneth E. Barner, Ph.D.
Chair of the Department of Electrical and Computer Engineering

Approved: _____
Babatunde A. Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: _____
Ann L. Ardis, Ph.D.
Senior Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Hui Fang, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Benjamin Carterette, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Chengmo Yang, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Stephan Bohacek, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

I would like to thank my parents for their consistent support and unconditional love throughout my entire life. I would like to give special thanks to my wife Yi Zhang. It is her who encourages me to pursue a challenging but fruitful life. Without her I cannot imagine I could achieve any of my accomplishment. Also, thank my kids for letting me being so joyful.

I would like to express my deepest appreciation to my advisor, Hui Fang. She opens a door for me which lets me to embrace a brand new world. Her diligence, persistence and hard working really inspire me a lot. It is such a honor to work with her and I will always appreciate such experience in my life.

I also thank other members in my dissertation committee Benjamin Carterette, Chengmo Yang and Stephan Bohacek. They gave me very useful and insightful suggestions and helped me to make the dissertation more solid and valid.

I would also thank my colleagues – Wei Zheng, Xitong Liu, Hao Wu, Yue Wang, Kuang Lv, Miguel Callejas, Monica Rodriguez, Wei Zhong, Hao Xu, Ye Wang, Zitong Cheng who gave me tremendous help. It was pleasure to discuss and work with them.

During my life at University of Delaware I have contacted many great people and I love here. I am and will always be a Fighting Blue Hen!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Information Retrieval (IR) is one of the most evolving research fields and has drawn extensive attention in recent years. Because of its empirical nature, the advance of the IR field is closely related to the development of various toolkits. While the traditional IR toolkit mainly provides a platform to evaluate the effectiveness of retrieval models, there are many emerging challenges that needs to be addressed using non-existing toolkits, e.g. teaching and research tools that scales at real world application, unified reproducibility evaluation system and also new applications such as contextual suggestion. In this thesis, we build various task-orientated IR toolkits in order to better address the new challenges.

First, the education oriented Virtual IR Lab (VIRLab) and Lucene-based Anserini is introduced in order to provide easy access to IR toolkits for both students and researchers. These toolkits can greatly reduce the instructor's work for teaching the IR courses especially the teaching of ranking models. VIRLab provides a web-based interface which enables the students to implement ranking models with just a few lines of API calls. It also includes many facilities such as automatic evaluation, search engine creation and the leader board for ranking models. Anserini is a command line interface (CLI) based toolkit built on top of Lucene. The advantage of Anserini lies in its capability of dealing with web-scale datasets and the utilities (e.g. multi-threaded indexing and streamlined TREC evaluation) that are essential to IR researchers.

Next, we propose a privacy preserving evaluation (PPE) framework in order to provide a general solution for the reproducibility study. In the framework, users would have different permission levels of accessing to the data – from choosing the ranking models from a list to leverage APIs to directly manipulate the index. We build a second level PPE system for a commercial dataset where users can leverage APIs

to implement their ranking models and the performance is automatically returned. We then introduce another instantiation of PPE framework – the Reproducible IR system evaluation (RISE) – in order to provide a unified evaluation system for the reproducibility study of IR ranking models. By using RISE, it is trivial to implement the ranking models and compare its performance with all existing models. We believe this could greatly reduce the redundant work of IR researchers on the unnecessary re-implementation of other ranking models. More importantly, the unified result RISE generates is the key to validate the utility of the proposed models.

Furthermore, we provide tools for analyzing the performance of existing ranking models for keyword queries. The best performing ranking functions such as BM25 and Dirichlet language model have been proposed for many years. Although there are some recently proposed ranking functions, their performances cannot easily surpass the old ones. Thus, it is interesting to investigate the reason behind this and also explore the performance upper bound if there is any. In this thesis, we first apply the gain/cost analysis in order to estimate the practical performance upper bound of single-term queries. We then identify the best subqueries for multiple-terms keyword queries by introducing several post-retrieval term relationship features. We argue that because the original queries do not follow the intuitions of the newly proposed the features, they do not achieve the better performances.

Last, we design an integrated contextual suggestion toolkit for contextual suggestion. The novelty of contextual suggestion mainly lies in its "zero query" property, meaning user does not need to submit query in order to get desired recommendations. Our toolkit consists of two components: a mobile application that can automatically detect the user's "context" (e.g. location and datetime); and the other component is essentially a recommendation system that can proactively suggests interesting venues based on user's current context and user's preference history. For the recommendation part, we investigate category-based and opinion-based user profile modeling approaches. Both methods work well on TREC and Yelp collections. Detailed analysis shows the advantage of opinion-based user profile modeling as it potentially answers

"why does the user like a place".

Chapter 1

INTRODUCTION

The past decades have witnessed the tremendous success of World Wide Web. People all over the world can now access to publicly available information via commercial search engines such as Google or Baidu with great ease. According to the online statistics[1], Google now (as of October 2016) can handle over 40,000 search queries every second on average, which translates to over 3.5 billion searches per day and 1.2 trillion searches per year worldwide. With such huge volume of search activities it is essential to keep improving the search infrastructures so that they can meet the user's information needs.

Information Retrieval (IR), usually used by academia in favor of its industrial counterpart search engine, is one of the most evolving fields and has drawn extensive attention in recent years. Although there are many research topics in IR that need sophisticated theoretical analysis, the experiments are always needed to validate the hypothesis proposed by the researchers due to the empirical nature of this fast evolving field. Traditionally, large amount studies related to the IR researches used toolkits mainly to measure the effectiveness and efficiency of the corresponding works. For example, the toolkits that measure the utility of the ranking model such as Indri and Terrier; the toolkits that measure the reading rate of a specific indexing method such as Indri. We can see that for some of the tools such as Indri has been used for many years since its first release (back in 2000). However, the fast pace of IR research and the emerging challenges that needs to be addressed require non-existing toolkits to address. For example, IR toolkit for teaching/learning, unified reproducibility evaluation

[1] http://www.internetlivestats.com/google-search-statistics/

system and other toolkits for new IR application such as contextual suggestion. In this thesis, we build various task-orientated IR toolkits in order to better address the new challenges and we will show more details in the following sections.

1.1 Tools for the IR Teaching/Learning

IR teaching and learning tools are essential for the spreading the IR techniques and thus get more recognition from the general crowds especially the college students. The tools also help the instructors to design the IR courses with more hands-on tasks for the students which greatly enrich students' knowledge base and their experimental skills. Unfortunately, current IR tools are mainly designed for more advanced researchers and thus are not quite suitable for the people without too much specialty of the field. We investigate some IR classes provided by colleges such as University of Illinois Urbana-Champaign and University of Delaware. The toolkits used those IR classes are the tailored version of Indri which was released years ago. Apparently, the IR community lacks the special version of the IR tools which is dedicated for teaching and learning the IR techniques. In this thesis, we propose two tools to address this problem. Specifically, Virtual IR Lab (VIRLab) [32], a web based IR system mainly aims for easy and fast evaluation of ranking models and Anserini [113], a Lucene-based tool targeted for providing web-scale search ability tool with Command Line Interface (CLI) and modular design. Designed as a web service, VIRLab provides an interface which enables the students to implement ranking models with just a few lines of APIs calls. It also includes many facilities such as automatic evaluation, search engine creation and the leader board for ranking models. All the functionalities are aimed to provide a system where the students can focus on the real important parts of the IR techniques without worrying too much about the system setup, scalability handling, etc. Anserini[2] is the other effort followed by VIRLab where the main concern is to provide a more advanced CLI based tool for both students and researchers. Anserini

[2] http://anserini.io

itself is built on top popular open source project Lucene[3]. In the first place, Lucene is not specifically designed for students or IR researchers. The documentation of Lucene is poorly maintained and this is, in our opinion, the main reason that prevents it from being widely adopted by the IR community. However, the merits of Lucene should not be ignored and this is where Anserini comes in. The advantage of Anserini lies in its capability of dealing with web-scale data collections such as ClueWeb12[4] of which the raw size is more than 7TB. This provides a good opportunity for people to have a better sense of real world application and how important of scalability to a real system. Anserini is also bundled with handy resources for IR research (e.g. queries and their judgments) and popular 3rd party libraries like Ranklib[5] and Deeplearning4J[6] which can be easily utilized by students and IR researchers to expand the current capability of the tool. Anserini's modular design is a perfect place for the users to configure and understand the corresponding components. For example, indexing option can be easily changed from porter stemmer to snowball stemmer so that the impact of such change can be monitored.

1.2 Unified Reproducibility Evaluation System

Another research endeavor is a long standing problem in evaluating the effectiveness of IR system[7]. For a typical IR evaluation system the ideal case is to have a unified testing environment which is responsible for everything related to the evaluation process except the ranking model part. That said, everything including pre-processing and indexing the documents, generating the ranking list, evaluating the results, the

[3] https://lucene.apache.org/

[4] http://lemurproject.org/clueweb12/

[5] https://sourceforge.net/p/lemur/wiki/RankLib/

[6] https://deeplearning4j.org/

[7] There are several aspects in IR system can be evaluated. We focus on the evaluation of effectiveness of the system. Specifically only the effectiveness of the ranking model is investigated

choice of evaluation metrics and interpretation of the performance, all should be under the same settings if one's purpose is purely compare the effectiveness of different ranking models. However, different research groups would like to maintain/use their favored toolkits such as Indri[8], Terrier[9] or ATIRE[10] where the settings of the toolkits are different enough to produce different ranking results [111]. Apparently the unified evaluation system is the very basis of comparing the effectiveness of ranking models and thus should be carefully treated. In this dissertation we propose two systems – privacy preserving evaluation platform (PPE) [112] and Reproducible Information retrieval System Evaluation (RISE) [111] in order to offer unified evaluation toolkits to the IR community for standardizing the comparison of ranking models.

It is well known that industrial data sets are valuable for the academic research since they reflect real world information needs in a more practical way. However, data from industry is normally protected by the owners of the data, e.g. companies like Google or Yahoo and can not be disclosed or redistributed due to intellectual property rights. Thus any experiments using the industrial data sets are often belong to some specific researchers and can not be reproduced by others. In order to better leverage the data sets from industry and bring the resource gap between the researchers who are inevitably not sharing the same resources in the world, we propose a privacy- preserving evaluation platform (PPE) for evaluation the information retrieval models. Specifically, we propose that there could be a hierarchical structure for this platform: each level exposures different aspects/information to the end users and thus the owner of the data can dynamically control the accessibility of the data. For example, the highest level of such system can just provide any available information of the data, e.g. the structure of the index or the APIs that manipulate the index, which enables the users to explore the data with greatest freedom. Then the second level could be hiding the details of

[8] http://www.lemurproject.org/indri/

[9] http://terrier.org/

[10] http://atire.org/index.php

the index but showing the detailed search results, e.g. ranking scores of the documents. The lowest level would be just the performance like MAP shown to the user without further information. Such hierarchical model can provide a viable solution to utilize the industrial data while the users do not necessarily have the permit to directly access the data, and thus is preferable. To test the proposed platform we evaluate some classic ranking functions such as BM25, Dirichlet language model, axiomatic model (F2EXP) using the PPE platform. By incorporating the diagnostic analysis [30] we are able to gain new insights about existing retrieval models. Although the experiments focus on the evaluation of basic IR models, the framework can be easily generalized for other tasks.

Followed by the idea of PPE we build RISE that instantiates the idea of PPE. The uniqueness and the advantage of RISE is that they offer centralized and controlled IR evaluation system which facilitates easy yet trusted comparison of retrieval models. With the help of RISE we are able to conduct a comprehensive reproducibility study for information retrieval models. In particular, we implement and evaluate more than 20 basic retrieval functions over 16 standard collections for IR research. Experimental results allow us to make a few interesting observations. We first compare the evaluation results with those reported in the original papers, and find that the performance differences between the reproduced results and the original ones are small for majority of the retrieval functions. Among all the implemented functions, only one of them consistently generates worse performance than the one reported in the original paper. Moreover, we report the retrieval performance of all the implemented retrieval functions over standard test collections for IR research. This is the first time of reporting such a large scale comparison of IR retrieval models. Such a comparison can be used as the performance references of the selected models.

1.3 Tools for Analyzing the Existing Models

With the unified IR evaluation system like VIRLab and RISE we have mentioned above we are able to compare ranking models with ease and trust. After a

comprehensive comparison of the most widely used ranking models [110, 111] we find that the optimum performances of some models [1,33,43,69,87,92,118] are quite similar on several data collections for IR research. The commonalities of those models are: (1) all of them are based on bag-of-terms document representation assumption. That is, terms in the document are independent with each other and the occurrence (or absence) of one term does not affect the occurrence (or absence) of any other terms, and (2) the models consist of basic signals (statistics) such as Term Frequency (TF), Inverted Document Frequency (IDF), Document Length Normalization (DLN) and other collection statistics [29]. With this finding some interesting questions here would be: it remains unclear whether we have reached the performance upper bound for such retrieval models. If so, what is the upper bound performance? If not, how can we do better?

To find the performance upper bound is quite challenging: although most of the IR ranking models deal with basic signals, how they combine the signals to compute the relevance scores are quite diverse due to different implementations of IR heuristics [29]. This kind of variants makes it difficult to generalize the analysis. Moreover, typically there are one or more free parameters in the ranking models which can be tuned via the training collections. These free parameters make the analysis more complicated. We can simplify the problem and just focus on single-term queries and study how to estimate the performance bound for retrieval functions utilizing only basic ranking signals. With only one term in a query, many retrieval functions can be greatly simplified. For example, Okapi BM25 and Pivoted normalization functions have different implementations for the IDF part, but this part can be omitted in the functions for single-term queries because it would not affect the ranking of search results. All the simplified functions can then be generalized to a general function form for single-term queries. As a result, the problem of finding the upper bound of retrieval function utilizing basic ranking signals becomes that of finding the optimal performance of the generalized retrieval function. We propose to use cost/gain analysis to solve the problem [10, 11, 28]. As the estimated performance upper bound of simplified/generalized

model is in general better than the existing ranking models, this finding provides the practical foundation of the potentially more effective ranking models for single term queries.

To extend the analysis from single-term queries to multiple-terms queries is not that obvious. However, some other clues can be utilized. Previous work [5, 52] have shown that verbose queries often contain unnecessary terms which are actually harmful to the effectiveness. Others [31] also found that this is also true for keyword queries for web scale collection. Our empirical experiments confirm that keyword queries can also be reduced in order to improve the performance for standard IR data collections. We argue that because the original keyword queries do not have some preferable properties that is why they can not achieve the best performance. In order to find such nice properties we propose a set of novel features that can better capture the relations among query terms, and then apply a learning-to-rank algorithm to rank the subqueries based on these new features as well as some existing ones. Specifically, the proposed features are query term proximity, the aggregated ranking scores of query terms, and the compactness and position of term tensors. The query term proximity features treat part of the query as phrases instead of separated terms and the intuition comes from the term dependency model [72]. They try to capture the "tight bond" between query terms. The aggregated ranking scores of query terms is originated from term frequency and inverted document frequency constraint [30] and it mainly investigate various high-level statistics from term scores (not the document scores) in the ranking list. The compactness and position of term tensors features view the term scores as tensors in the multi-dimensional space and measure the spatial properties of the tensors cluster. We rank all subqueries for each original keyword query in standard IR data collections using the state-of-the-art learning-to-rank algorithm LambdaMART. The subquery ranking results show that we can identify the best subqueries more effectively. We further do intensive feature importance analysis. The results verify the utility of our proposed features.

1.4 Contextual Suggestion Tool

Last, we design an integrated contextual suggestion toolkit for the automatic detection of the users "context" (e.g. location and date) via mobile application and then proactively suggests interesting venues to the user based the context. The contextual suggestion is a new area of the active IR research and the most notable feature of contextual suggestion is its "zero query" property – to recommend interesting venues to the users based on contextual information such as geographic location, temporal information and user's activity history but without user's input query. For this task, we first build a mobile phone application which can silently track the current context of the user. We then recommend the interesting places to the user based on the detected context.

User profiling is the key component to effectively rank candidate places with respect to a user's silent information need. In order to model use profile we first propose to leverage the category and description information about the places in user's activity history to construct user profiles [105]. The advantage of such approach is the ease of computation and the satisfactory results [27]. We further find that using category or description to build a user profile is not enough: category of places is too general to capture a user's underlying needs; while the text description of a place is too specific to be generalized to other places. In other studies [106–109, 114] we leverage opinion, i.e. opinion ratings and the associated text reviews, to construct an opinionated user profile. By doing like this we aim to explain "why the user likes or dislikes the suggestion" instead of simply recording "what places the user liked or dislike" in the search history. The problem of this approach is that on-line opinions are notoriously skewed as only very small number of people post their opinions. To address this data sparsity challenge we propose to also include the opinions from similar users as the current user to construct the profile of current user. The assumption here is that users with similar ratings have the similar reasons of giving the rating. By modeling the candidate places in the similar fashion the similarity between user profile and candidates profile is used to rank the candidates. We tried different representations

of the text reviews when modeling the profiles. We further apply Learning-to-Rank (LTR) method to the similarity scores for the ranking method. Experiment results on standard data collections and a self-crawled Yelp collection validate the effectiveness of the method.

1.5 Summary

IR community needs a set of new tools to address the emerging challenges. In the following of this thesis we will be introducing more details on the tools we develop for various tasks.

The rest of the thesis is organized as follows. First, we discuss some general background of the thesis and the previous work in Chapter 2. In Chapter 3, we discuss the IR teaching and learning tools VIRLab and Anserini – their advantage in providing the real world experience to the users and the typical use cases. We describe our effort on the unified evaluation systems in Chapter 4. There we will have detailed illustration of our proposed systems, namely PPE and RISE. We explain how we build tools to analyze the performance upper bound for single term queries in Chapter 5. Later in that chapter we continue to introduce how we identify key features to reduce the multiple-terms keyword queries. In Chapter 6, we continue to explore the tools for new IR application – the contextual suggestion, where we first build a mobile application to capture user's location and temporal information. We then build a recommendation tool to suggest interesting places to the user based on user's preference history. We finally summarize the thesis and discuss the future work in Chapter 7.

Chapter 2

BACKGROUND AND RELATED WORK

In this thesis, we build the tools and systems for various IR tasks in order to better address the new challenges of the emerging problems. Specifically, four sets of tools are proposed: tools for IR teaching and learning; tools for reproducibility study and the evaluation; tools for analyzing the performance of existing ranking models and the tools for the new IR application i.e., contextual suggestion. In this chapter, we will describe the necessary background and the key concepts for understanding the thesis. We will also describe the previous work in the related research fields.

2.1 TREC and TREC Collections

Most IR research deals with data collections no matter which specific problem is been explored. There are several standard data collections we have mentioned in Chapter 1. Since for most of the work, we mainly focus on the most basic problems in IR, e.g. the efficiency and the effectiveness of ranking models as well as the indexing over the standard TREC Ad-hoc/Web collections, It is essential to introduce TREC conference and TREC Ad-hoc/Web collections.

2.1.1 TREC

TREC stands for Text RetriEval Conference[1] and is co-sponsored by the National Institute of Standards and Technology (NIST) and U.S. Department of Defense. The first TREC was held in 1992 as part of the TIPSTER Text program. Its purpose was to support research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies. In particular, the TREC workshop series has the following goals:

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

- to encourage research in information retrieval based on large test collections;
- to increase communication among industry, academia, and government by creating an open forum for the exchange of research ideas;
- to speed the transfer of technology from research labs into commercial products by demonstrating substantial improvements in retrieval methodologies on real-world problems; and
- to increase the availability of appropriate evaluation techniques for use by industry and academia, including the development of new evaluation techniques more applicable to current systems.

There are several tracks in TREC such as Ad-hoc track [41], Robust track [98, 99], Web track [22], Million Query track [15], Microblog track [62, 63], Contextual Suggestion track [25, 26], Session Track [12, 13], etc. Apparently, different tracks have different purposes. Our work mainly relates to Ad-hoc track, Web track and Contextual Suggestion track.

Ad-hoc track mainly refers to the tracks in earlier years whose main aim is to improve the effectiveness of general search, e.g. random daily searches. Web track is very similar to Ad-hoc track in terms of the formatting – it mimics the general searches from the users. However, the data sets used in Web track is in general larger than that of Ad-hoc track in order to reflect the fast-growing of the data in the real world. Contextual Suggestion track is different from Ad-hoc track and Web track. There are no queries in this task to model the dynamic needs of mobile search. Users' preference histories instead of the queries are provided and the system is expected to recommend interesting places and events based on user's current context.

2.1.2 TREC Ad-hoc/Web Collections, Topics, Judgments and Evaluation Metrics

One of the most valuable stuff of TREC is its data collections, the queries, and the corresponding judgments for the collections. They together provide a standard way to evaluate the systems, models, algorithms submitted by different research groups all over the world.

In our thesis, we mainly use the Ad-hoc track data collections and the Web track data collections. Disk1&2, Disk4&5, AQUAINT are the data collections for Ad-hoc track. They are relatively small (less than 5GB per collection) and old (were used in 2005 TREC or before). The contents are mainly news articles and government records. WT2G, WT10G, GOV2, ClueWeb09, ClueWeb12 are for Web track and all of them were crawled from the web. WT2G, as its name suggests, is only 2GB. WT10G is 10GB. GOV2 is crawled from .gov domain and is over 270GB. ClueWeb09 and ClueWeb12 are the much larger data sets recently crawled. ClueWeb09 is over 5TB uncompressed and ClueWeb12 is over 7TB uncompressed. There are two sub-collections for ClueWeb collections – ClueWeb09B and ClueWeb12B where only the English pages are kept and the non-English pages are not part of them. ClueWeb09B and ClueWeb12B are roughly 1/10 in terms of size compared with the full collections.

Topics are essential the queries. For Ad-hoc and Web track, some queries contain three parts: title, description, and narrative. Title query is very short, about 2 to 6 words. Description query is longer and the narrative is the longest with much more details such as the intention and the expected results of the query.

Judgments contain which documents in the collection are relevant to a specific query. There are basically two types of judgments: binary and numeric. For binary judgment the document is labeled as relevant (usually "0") or non-relevant (usually "1"). For numeric judgment, numbers are assigned to documents indicating the different level of relevant, e.g. highly relevant, relevant and non-relevant.

Evaluation metrics are mainly used to as a single point indicating how good is the model or system. Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (nDCG), Precision, Recall are the most commonly used ones. MAP is mainly for binary judgment while nDCG is for numeric judgment. Precision and Recall are often followed by a number indicating we only judgment till a specific position in the ranking list. For example, P@5 means we only look at the top 5 ranked documents in the ranking list and the metric is just for them.

2.2 Typical Ranking Models and Index Structure

We would like to introduce the typical ranking models such as BM25 [87] and Dirichlet Language Model [82] and the key information stored in an index. Some ranking models will be used for the entire thesis so it is better to introduce them before actually use them. BM25 and Dirichlet Language Model are the most widely used ranking models in IR research. BM25 was proposed by Robertson et al. back in 1994 in TREC-3. Dirichlet Language Model was firstly proposed in 1998. Albeit the mentioned models are old, they are still among the most effective ranking models [111]. These models are based on bag-of-terms document representation assumption where the terms are seen individually and there is no relationship between them. Although this assumption may be far away from the truth, the ranking models based on this assumption have decent performance. Other popular bag-of-terms ranking models include the Divergence from Randomness model [1] and Axiomatic Models [33]. In chapter 3, we will introduce the different index for the data collections and we would like to provide some basic knowledge about the index and the relationship between the index and the ranking models. When we build the index for a document (or a set of documents) we are basically extracting the information from the document. We will introduce the most important ones here: term counts, positions and document vector. Term counts are the most basic information and are basically the number of occurrence of each term in the document. Most ranking models which are based on bag-of-terms, such as BM25 and Dirichlet Language Model need term counts in order to work. Positions record the position of each term in the document. This is required for proximity query where the distance between query terms is needed. The structure of most indexes is the inverted posting lists where the terms are the entries of the posting lists and there is no information about a specific document, e.g. which terms are in a document. But index can also have document vector where the terms and their counts and positions of a document are also stored. The document vector is essential for relevance feedback where we look at the top ranked documents and compare the query terms with the document vector.

2.3 IR Toolkits for Teaching and Research

Tools for teaching and research are always important for any research field. There are various efforts from the related fields such as machine learning, signal processing [45] or natural language processing. There have been significant efforts on developing various single-machine tool and web services for IR teaching and research.

We first introduce some long-lived and popular tools for IR researchers. Indri[2] is probably the most popular and one of the oldest toolkit for IR research community. The main purpose of Indri is to provide the indexing and searching functionality mainly focus on the inference network with language models. Galago can be seen as the upgraded version of Indri which is written in Java. The emphasize of Galago is to provide the similar functionality with Indri while improving the efficiency. Terrier[3] is another excellent tool for efficient indexing and retrieval. Lucene[4] is an open source project built for easy deployment of the real-world search application. Some biggest advantages of Lucene include its capability of dealing with web-scale data set and the general acceptance of from people. There are some other tools which also provide the basic functionality of indexing and retrieval over data collections such as ATIRE[5].

We build Anserini on top Lucene in order to leverage its merits since Lucene is not intrinsically for IR research. Anserini ships with the following functionalities: multi-threaded indexing, TREC topics and judgments for streamlining the evaluation and utilities such as relevance feedback. These functionalities are in hope of bridge the gap between Lucene and IR researchers to let them better understand the advantage of Lucene when dealing with large web-scale data collections.

Move the evaluation from the single machine tool to web service or Evaluation-as-a-Service (EaaS) [60, 61, 86] and make it automatic is a new trend. The trend is

[2] https://www.lemurproject.org/indri/

[3] http://terrier.org/

[4] https://lucene.apache.org/

[5] http://atire.org/index.php?title=About_ATIRE

closely related to the reproducibility study. The SIGIR 2015 Workshop on Reproducibility, Inexplicability, and Generalizability of Results (RIGOR) [44] is one of the venues that encourage the study of reproducibility. Their reproducibility challenge invited developers of 7 open-source search engines to provide baselines for TREC GOV2 collection. Trotman et. al. [96] and Muhleisen el. al. [74] have also tried to reproduce retrieval results for IR models, but the number of retrieval functions and the number of collections used in these studies (1 function 1 collection for [96] and 9 functions 10 collections for [74]) are not as large as what we studied in this paper. Lin et al. [58] proposed an open-source IR reproducibility challenge where they split the IR system into pieces of components such as two kinds of tokenization methods and four different IR toolkits. By easily configuring different combinations of these components, we can have a partially filled matrix indicating the performances of specific combinations of the components. Such transparent experiment set up makes it possible to have a better understanding about the impact of different components. Gollub et al. [38] described a reference implementation of their proposed IR evaluation web service which bears the important properties like web dissemination and peer-to-peer collaboration. Hanbury et al. [39] reviewed some of the existing automated IR evaluation approaches and proposed a framework for web service based component-level IR system evaluation. Lagun and Agichtein proposed a web service, which enables large scale studies of remote users [53]. Their system focused on providing a platform that reproduces and extends the previous findings on how users interact with the search engine especially the search results. Xitong et al. [67,68] provided a system that can benefit the teaching of IR by introducing the entity related tasks.

We have developed three systems: Virtual IR Lab (*VIRLab*), Privacy Preserving Evaluation (*PPE*) and Reproducibility Ir System Evaluation (*RISE*). VIRLab [32] provides a web service for users to implement retrieval functions. It is mainly designed to facilitate teaching IR models. PPE is designed mainly for IR researchers in academia to leverage the private industrial data sets. The system hides the data collection (or the index) from the users, which avoids the re-distribution of the data collection. RISE

is also designed as a web service to provide a unified interface for the users to evaluate their models/algorithms. The uniqueness of *RISE* system is that it is specifically designed to facilitate the implementation and evaluation of retrieval functions. It hides the details about collection processing and evaluation, and enables users to focus on only the implementation of retrieval models. Because of its flexibility, we are able to implement and compare a wide range of retrieval functions that were not implemented in any other open-source toolkits. Second, our reproducibility study includes more retrieval functions and more data collections. The ultimate goal of the *RISE* system is to provide a complete set of benchmark results of IR models.

2.4 Tools for Analyzing the Ranking Models for Keyword Queries

2.4.1 Performance Upper Bound of Single-Term Queries

Although there are lots of effective ranking models proposed by researchers, there are fewer studies dedicated to the theoretical analysis of their performances upper bound. One related domain is the constraint analysis [29] which proposes formal constraints that a reasonable ranking model should bear. Examples of the constraints including how should a ranking model incorporate TF, how to regulate the interaction of TF and DL, how to penalize long document in the collection, etc. The constraint analysis provides a general guide of how a reasonable ranking model should be designed. Our work further explores this direction by providing the practical performance upper bound as well as the optimal parameters which helps to fine tune the constraint theory.

Our estimation method is mostly inspired by the RankNet [10,11] and the LambdaRank [10,28] which are successful in the learning to rank domain. In their works they apply the pair-wise documents comparison for a specific query which is also adopted by our work. However, we did two different things in our work: (1) the aforementioned techniques apply neural network as the underlying model while we follow the rationale proposed by some classic ranking model, i.e. the ranking score should be positively correlated with TF and inversely correlated with DL, to find the local optimum of the generalized ranking models. (2) we aim to optimize MAP instead of NDCG and we

proposed a simplified equation for calculating the difference of MAP if two documents are swapped in the ranking list which can make the analysis more efficiency. There is another work which indeed directly optimizes MAP called SVMMAP [116]. SVMMAP is actually another learning to ranking algorithm based on support vector machine. It performs optimization only on a working set of constraints which is extended with the most violated constraint at each optimization step. Taylor et al. [95] used the cost analysis to predicate a family of BM25 ranking models. They however did not apply the gain analysis which has shown to be superior in our experiments.

2.4.2 Multiple-Terms Keyword Queries Reduction

Reducing verbose queries to shorter queries has been intensively studied in recent years. Most previous work involves in generating the features for either the original query Q, the subquery q or groups of terms. Basically there are several features categories:

Statistical Features TF-IDF based features are the most widely used set of statistical features which include various statistics such as collection TF [8], IDF [52], residual IDF [23], TF in matching Wikipedia titles [47], count of passages containing the subquery [80, 103] etc. Other popular features include simplified clarity score [23, 52] and mutual information (MI) between query terms [50, 51, 104], domain specific dictionaries based features such as whether the term indicating a brand [104].

Query Features Query features are only based on the query itself and no collection context is involved. Similarity Original Query measures the cosine similarity between TF-IDF vectors of each subquery and the original query [52]. Presence of Stop Words [5, 80] computes the ratio of stop words in subquery. IsRightMost and IsLeftMost [104] are the features that capture the position of the subquery in the original query.

Term Dependency Features These features capture the dependencies between query words. Park et al. [80] proposed four types of dependencies among query terms: parent-child, ancestor-descendant, siblings and c-commanding. The final features include the

number of dependent clauses in the query; the ratio of the dependent term pairs which have parent-child; ancestor-dependent, siblings, and c-commanding in the query.

Post Retrieval Features These features are based on the ranking results of sub-queries. Typically, these features are expensive to compute but they have been proven to be effective. Query-document Relevance Scores [5, 18] are the LambdaRank and BM25 scores of top K documents, the click through counts of top K documents and the page-rank scores of top K documents. Query scope [52] of a subquery is the size of the retrieved document set relative to the size of the collection. Weighted Information Gain [8] is the difference of the retrieval quality by comparing the state where only the average document is retrieved to the state where the actual results are observed. Query drift among results [23] is a set of features which include the standard deviation of the ranking scores at 100 documents, the maximum standard deviation in the ranking list, etc.

Our proposed features are all *post-retrieval features* and some of them share the similar ideas with previous studies, e.g. term proximity based features are inspired by the term dependency features mentioned above. Different from the previous work, we are interested in the document score of the proposed term proximity models and various properties of the term scores.

A large number of research efforts have been made towards combining the features using a classification or a regression model. The classification problem is equivalent to pick the best subquery and it typically decides whether a term in the original query should be included in the best subquery. The regression problem is to learn a weight for each term denoting its importance score or to learn a weight for a sub-query; the top terms or the subquery with highest weight is then chosen. RankSVM [5,52,79], decision trees, AdaBoost, logistic regression [104] are popular classification models while random forests [5] is the most popular regression model. We adopt the classification model in our work where we apply the LambdaMART to the features from all subqueries and the model learns which subquery should be the best subquery.

2.5 Contextual Suggestion

The problem of contextual suggestion was first introduced at TREC in 2012, and the track has been running in the past four years [24–27]. Although the details of the track varied, the task remains the same. Given a user's preferences on a set of example suggestions and a context, track participants are expected to return a ranked list of new suggestions that are likely to satisfy both the user preferences (based on their preferences on the example suggestions) as well as the contexts such as geotemporal locations. Each example suggestion includes a title, description and an associated URL. For each user, we know their preferences on part or all of the example suggestions.

Most TREC participants retrieved candidate suggestions from various online services such as Google Place or Yelp based on the geographical context and then use some heuristics, e.g. nightclub will not be shown if the temporal context is in the morning, to filter out the suggestions that do not match the temporal contexts [25,27]. After that, the task is to retrieve useful suggestions based on user preferences. Most participants formulated the task as a content-based recommendation problem. For example, Hubert and Cabanac [46] used the terms in the description as the profile of the places. They use the web contents of the candidate places to compare with the example places in order to get rank the candidates. Yang et al. [115] used the proportion number of the places in each category to select the places among categories. Yang et al. [105] applied the hierarchical category structure to compute the similarity between examples and candidates. Rao and Carterette [85] also leveraged the TREC official description and other text description crawled from the open web to model the problem as a general text retrieval problem. McCreadie et al. [71] tried to extract the business hours, location of the business, tf-idf value of description terms in Wiki Travel as the features to feed a learning-to-rank algorithm. A common strategy adopted by top-ranked participants of TREC is to estimate a user profile based on the example suggestions and then rank candidate suggestions based on their similarities to the user profile. The basic assumption is that a user would prefer suggestions that are similar to those example suggestions liked by the user.

There are some studies that used terms from the description of the places or the web pages of the example suggestions to construct user profiles, and then various similarity measures are used to rank the candidates [105, 114]. A few studies also explored the use of category information for user profiling and candidate ranking. For example, Li and Alonso [57] utilized the accumulative category scores to model both user and candidate profiles, and then use the full range cosine similarity between the two profiles for candidate ranking. Li et al. [56] leveraged how likely each popular category is liked/disliked by users to construct user profiles, and the candidate ranking is to favor suggestions from a user's favorite categories. McCreadie et al. [71] proposed to rank the candidates by comparing two trees of finer-grained categories between user profile and candidate profile using a tree-matching technique. Diversification is then applied so that the categories of top ranked candidates are normalized. Yates et al. [115] proposed to recommend the candidates which are proportional to the number of example suggestions in each category. Koolen et al. [49] applied a similar method with a major modification of retrieving the category information from Wikitravel[6].

Although we also use category and description of example suggestions to build user profile, we propose to leverage text reviews about the example suggestion to estimate the user profile which is unseen from previous works.

2.5.1 Recommendation Systems

The problem of contextual suggestion is also similar to collaborative filtering [93]. Collaborative filtering assumes that similar users would share similar ratings, and focuses on predicting the user rating based on such an assumption. It often requires a large number of past user preferences to be more accurate and sometimes it may suffer from data sparsity problem which is known as the cold start problem [89]. In order to solve the data sparsity problem, reviews were incorporated to improve the performance. Hariri et al. [40] inferred the context or the intent of the trip by analyzing reviews. In particular, they used latent Dirichlet Allocation to identify the topics from the

[6] http://www.wikitravel.org/

reviews, and the final ranking scores are generated based on both the context scores as well as the scores generated by traditional collaborative filtering methods. Jakob et al. [48] proposed to cluster the features and then apply natural language processing techniques to identify the polarity of the opinions. A few studies also focused on leveraging Location Based Social Network to solve the data sparsity problem. Noulas et al. [76] applied random walk based on latent space models and computed a variety of similarity criteria with venue's visit frequencies on the location based social newtowkr. Bao et al [6] proposed to first constructing a weighted category hierarchy and then identify local experts for each category. The local experts are then matched to a given user and the score of the candidate is inferred based on the opinions of the local experts.

Our work is also related to other studies that utilized reviews to improve the performance of recommendation systems. Raghavan et al. [84] proposed to use the helpfulness, features from the text reviews and the meta-data (average rating, average length of text reviews and etc.) of the opinions to train a regression model in order to generate a quality score for each opinion. The quality score is then incorporated into the probabilistic matrix factorization as an inverse factor which affects the variance of the prediction from the mean of the factor model. Levi et al. [55] extended this study and analyzed the review texts to get the intent, features and the ratings for each feature. Qumsiyeh and Ng [83] explored the aspects in the reviews and computed the probability of each genres (categories) in each rating level. Their work is limited to the applications in multimedia domains, and the genres of each type of media is pre-defined.

Our work is different from these previous studies in the following aspects. First, our focus is to directly use reviews to model user profile while previous studies mainly used reviews to predict the rating quality or the user intent. Second, existing studies on collaborative filtering were often evaluated on only specific applications, e.g., movies, hotels, and it is unclear how those methods could be generalized to other domains. In contrast, our proposed method is not limited to any specific domains and can be applied to a more general problem set up.

Chapter 3

TEACHING/LEARNING TOOLS FOR INFORMATION RETRIEVAL

IR teaching and learning tools are essential for the spreading the IR techniques and thus get more recognition from the general crowds especially the college students. The tools also help the instructors to design the IR courses with more hands-on tasks for the students which greatly enrich students' knowledge base and their experimental skills. Unfortunately, current IR tools are mainly designed for more advanced researchers and thus are not quite suitable for the people without too much specialty of the field.

In this thesis, we first propose Virtual IR Lab (VIRLab) [32], a web based IR system mainly aims for easy and fast evaluation of ranking models. VIRLab lets the users of the system to implement their ranking functions using the system provided APIs for manipulating the index. The implemented ranking functions are automatically evaluated and the performance is returned to the users. VIRLab has other utilities such as leader board which compares the ranking functions submitted by the users, search engine creation which enables the users to create a demo search engine, detailed comparison between ranking functions where the graphical tools can be used to investigate the ranking difference between two functions side by side.

Anserini aims to enable IR researchers to leverage the power of Lucene but with much less burden on exploring Lucence from scratch. Specifically, Anserini provides a thin wrapper that can allow users to use high-level APIs to pick and combine some underlying code pieces from Lucene to accomplish their tasks. Moreover, the modular design of Anserini makes the implementation of new components more flexible and faster. In addition to search, Anserini can also support new functionalities such as learning-to-rank and word embedding. Finally, shipped with all TREC ad hoc/web

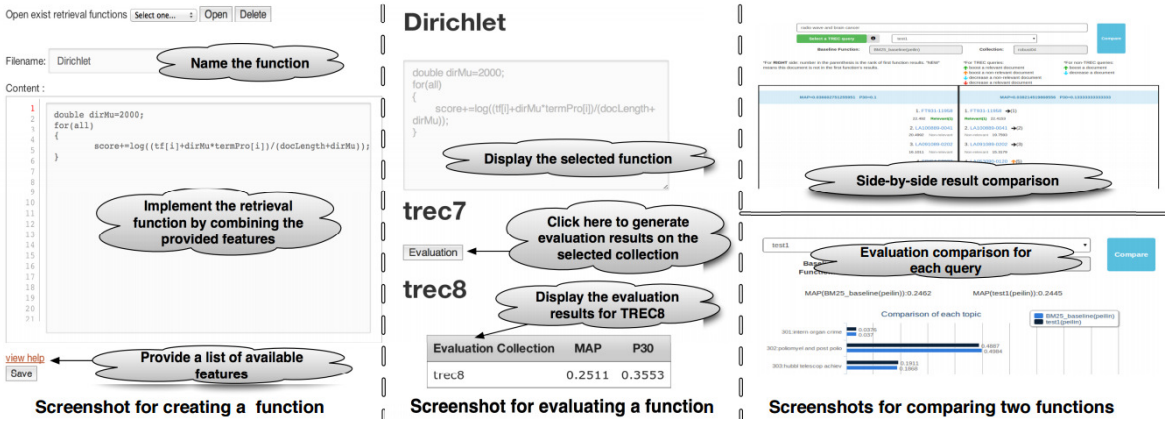


Figure 3.1: Screenshots of function creation (left), function evaluation (center) and function comparison (right)

track topics and judgments, Anserini provides a convenient way to obtain such publicly available resources for IR researchers and greatly reduces the unnecessary replicated work. When comparing the Anserini with other academic IR toolkits, Anserini has been shown to be more efficient in terms of both indexing and retrieval while maintaining comparable performance in terms of effectiveness.

3.1 Virtual IR Lab(VIRLab)

In this section we describe our efforts on developing a web-based tool for IR researchers and students to study retrieval functions in a more interactive and cost-effective way. Our developed Virtual IR Lab (VIRLab) is our first attempt of making a unified evaluation system and it can offer the following functionalities:

- *Easy implementation of retrieval functions*: Users only need to write a few lines of code through a Web form to combine statistics retrieved from the indexes without worrying about how to access the indexes. The code will be automatically checked for syntax errors and translated to an executable, which will be used for ranking documents, by a dynamic code generator.
- *Flexible configuration of search engines*: Users can configure a search engine by selecting a retrieval function and a test collection. Multiple search engines can be easily created at the same time. The users can either submit their own queries or select queries from a set of topics associated with the corresponding document collection. Moreover, the users can also compare the search results of two search engines side by side to figure out their ranking differences.

- *Tight connections among implementation, evaluation and result analysis*: After creating a retrieval function, the users can evaluate its effectiveness over a few provided test collections by simply clicking a button. If a retrieval function contains multiple parameter values, the users may select to evaluate all of them. If a search engine is configured using an existing test collection with relevance judgments, the official queries and judgments will be displayed so that the users can easily analyze the search results to figure out when the search engine fails and why.
- *Performance comparison through leader-boards*: A leader board is created for each collection so that the most effective 10 retrieval functions are displayed. Users can see how their retrieval functions are compared with others, and they can also leverage the comparison functionality described earlier to figure out how to revise their retrieval functions to improve the performance.

Figure 3.1 shows the screen-shots of three major functionalities including creating a retrieval function, evaluating the function and comparing the results of two functions. We now provide more details about these functionalities. The front end of the system is a web interface that allows users to create retrieval functions. Specifically, a user can implement a retrieval function by simply combining multiple features (i.e., collection statistics) from a provided list based on C/C++ syntax. As an example, the left part of Figure 3.1 shows how the Dirichlet prior retrieval function is implemented. Moreover, instead of specifying a single parameter value, the users can also specify a set of values for retrieval parameters, and then the system will automatically create a group of functions with these parameter settings. Once a retrieval function has been created, the user can select test collections and evaluate the effectiveness of the retrieval function over the collections (as shown in the middle part of Figure 3.1).

The front end also enables users to use or evaluate the retrieval function through a web-based search interface. The user first needs to create a search engine by selecting a retrieval function and a document collection. After that, the user can either enter her own query or select a query from existing test collections when queries are available. If the query is from the test collections, we will display not only search results but also the relevance judgment of these results as well as the evaluation results for the query. This feature would allow users to easily see when their search engines fail or

succeed and encourage them to identify the problems and try to fix them by changing the retrieval function. Moreover, we also empower users to compare the search results of two search engines side by side so that they could analyze them and identify how to revise one of the search engines accordingly. The right part of Figure 3.1 shows the screen shot of this functionality.

To promote controlled experimental study of IR, we generate a leader-board to report the best performed retrieval functions for each collection. Users can compare the results of the best system with their own retrieval functions through both side-by-side search result comparison and quantitative evaluation comparison.

The back end of the system includes several basic components such as indexer, ranker and evaluation script. The indexing process is done off line. Several standard TREC ad hoc collections have been indexed and ready for users to choose from. The ranker is determined by the retrieval function that the user provided through the front end.

As our first attempt, VIRLab is suitable for general purpose of IR research or study. However, if we want to deploy a large scale of reproducibility study then we need a more advanced system where users can collaborate with each other and focus on the implementation and the evaluation of the retrieval functions. We will introduce the other effort RISE in the next section.

3.2 Anserini

3.2.1 Introduction

Information retrieval researchers have a long history of developing, sharing, and using software toolkits to support their work. Over the past several decades, various IR toolkits have been built to aid in the development of new retrieval models, to test hypotheses about information seeking, and to validate new evaluation methodologies. As the field moves forward, IR toolkits are expected to keep up with emerging requirements such as the ability to handle large web collections and new data formats. The growing complexity of modern software ecosystems and the resource constraints felt

by most academic research groups make maintaining open-source toolkits a constant struggle.

Most IR toolkits developed by academics, such as Indri,[1] Galago,[2] and Terrier,[3] were primarily designed to facilitate evaluation over standard test collections from evaluation forums such as TREC, CLEF, NTCIR, etc. In many cases, scalability took a back seat to efforts around improving retrieval models, and thus these systems often struggle to scale to modern web collection. As an example, the ClueWeb12 collection[4] contains 733 million web pages, totaling 5.54 TB compressed (or 27.3 TB uncompressed). The standard practice for working with this collection, as exemplified by the infrastructure built for the TREC 2014 Session Track [14], is to separately index partitions of the collection and then build a distributed architecture that integrates results from each partition. In general, working with web-scale collections using existing academic IR toolkits is time- and resource-intensive, even for basic tasks.

With the exception of a small number of companies (e.g., commercial web search engines), the open-source Lucene system[5] and its derivatives such as Solr and Elasticsearch (for convenience, we simply refer to as "Lucene" collectively in this thesis) have become the *de facto* platform for deploying search applications in industry. Examples include LinkedIn, Twitter, Bloomberg, as well as a number of online retailers and many large companies in the financial services space. Despite its undeniable operational success, a large user base, and a vibrant community of contributors, Lucene is not well suited to information retrieval research. For many reasons, including poor documentation of system internals and a number of unintuitive abstractions, Lucene is not as widely used for research as academic toolkits such as Indri or Terrier.

---

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

[2] http://www.lemurproject.org/galago.php

[3] http://terrier.org/

[4] http://www.lemurproject.org/clueweb12/

[5] https://lucene.apache.org/

In this thesis, we describe our efforts in developing a new open-source information retrieval toolkit called Anserini that builds on Lucene.[6] We aim to bridge the gap described above that separates the practice of building real-world search applications from information retrieval research. Anserini provides wrappers and extensions on top of core Lucene libraries that allow researchers to use more intuitive APIs to accomplish common research tasks. Our initial efforts have focused on three functionalities: scalable, multi-threaded inverted indexing to handle modern web collections, streamlined IR evaluation for *ad hoc* retrieval on standard test collections, and an extensible architecture for multi-stage ranking. Anserini ships with support for standard TREC test collections, providing a convenient way to replicate competitive baselines "right out of the box", supporting the community's aspirations toward reproducible results [2, 32, 35, 59, 100, 111].

We experimentally evaluate the efficiency and effectiveness of Anserini on a number of standard test collections. In terms of indexing performance, it is able to handle the largest research web collection available today with ease on a single modern server. We observe better indexing performance compared to Indri, a popular choice among researchers today. In terms of retrieval, we also find that Anserini is not only faster than Indri, but returns rankings that are comparable in quality. In other words, Anserini is faster and just as good. We present the case that Anserini should be adopted as the toolkit of choice for information retrieval researchers.

3.2.2 Motivation

Despite its popularity in industry and broad adoption for operational search deployments, Lucene remains under-utilized in information retrieval research. We begin with some high-level discussions of why we believe this might be so to motivate our efforts in building Anserini.

From the very beginning, Lucene was written for "real world" search applications, not with researchers in mind. For the most part, its developers targeted an

[6] http://anserini.io/

audience that mostly used search engines as black boxes, as opposed to researchers that required access to ranking internals such as scoring models, mechanisms for postings traversal, etc. Because of the target user population, documentation for Lucene internals has always been quite poor, especially in keeping up with the relatively rapid pace in which the developer community has been releasing improved versions of the software. Access to these internals is exactly what information retrieval researchers need for their studies, and therefore poor documentation has been a barrier to entry.

To further compound this issue, the internal APIs in Lucene are not organized in a way that would be intuitive to most IR researchers, with class names that are not indicative of functionality and many levels of indirection. This is not an issue for "black box" users of Lucene, but presents a hurdle for information retrieval researchers who desire access to system internals. As an example, the code to open up a Lucene index and to traverse postings programmatically (without invoking the scoring function) is unnecessarily complex and involves dispatching to several intermediate classes along the way. Some researchers have the impression that Lucene is difficult to use, and indeed there is some truth to this, especially with respect to low-level abstractions.

Another side effect of Lucene's focus on "black box" search is that it has severely lagged behind in the implementation of modern ranking functions. For the longest time, the only scoring model available was an *ad hoc* variant of tf-idf. BM25 was not added to Lucene until 2011,[7] more than a decade after BM25 gained widespread adoption in the research community as being more effective than tf-idf variants. This lag in adopting "research best practices" has contributed to the perception that Lucene is not effective and ill-suited for information retrieval research. However, this perception is no longer accurate today. Lucene comes with implementations of modern baseline retrieval models, and we show that the effectiveness of Lucene's implementations is at least as good as those offered by academic IR toolkits (see Section 3.2.4).

Finally, because Lucene is written in Java, there is sometimes the perception

[7] https://issues.apache.org/jira/browse/LUCENE-2959

that it is slow and inefficient, particularly when scaling up to modern web collections. Developers often point to the managed memory environment of the Java Virtual Machine (JVM) as not being conducive to efficient low-level implementations of search engine internals. We experimentally show that this is definitely not true (see Section 3.2.4). The open-source community has devoted substantial effort to optimizing the performance of Lucene and taking advantage of today's multi-core processors. It is capable of handling large web collections on a single server with ease.

The goal of Anserini is to align the research and practice of building search applications with research in information retrieval. Colloquially speaking, our toolkit aims to smooth out the "rough edges" around Lucene for the purposes of information retrieval research. It is not our goal to replace or reimplement Lucene, but rather to facilitate its use as a research toolkit by presenting as gentle a learning curve as possible to newcomers.

3.2.3 Main Components

Anserini Components fall into two categories: *wrappers* and *extensions*. Wrappers provide APIs that leverage core Lucene library components to accomplish specific tasks. They are tightly integrated with "core" Lucene and in some cases, represent custom implementation of existing Lucene APIs. Extensions, on the other hand, are components that are distinct from Lucene and more loosely coupled: these may represent our own implementations or connectors to third-party libraries.

Multi-threaded indexing (*wrapper*). Inverted indexing is one of the most fundamental tasks in information retrieval and the starting point of many research studies. In working with large web collections, it is imperative that indexing operations are efficient and scalable. While academic researchers have attempted to address this issue via MapReduce and related frameworks [16, 64], these solutions impose the burden of requiring clusters and additional software infrastructure.

Lucene supports multi-threaded indexing, and as we experimentally show (Section 3.2.4), it is able to scale up to large web collections on a single commodity server.

Table 3.1: Indexing performance of Anserini on smaller collections using 16 threads on a modest commodity server. Index with term counts only and with term counts and positions are included.

			Anserini (count)		Anserini (pos)	
Collection	docs	terms	time	size	time	size
Disk12	742k	219m	00:01:27	199MB	00:01:30	512MB
Disk45	528k	175m	00:01:18	166MB	00:01:28	423MB
AQUAINT	1.03m	318m	00:01:54	305MB	00:01:44	734MB
WT2G	246k	182m	00:02:37	143MB	00:02:54	437MB
WT10G	1.69m	752m	00:05:24	708MB	00:05:21	2.9GB
Gov2	25.2m	17.3b	01:16:18	11GB	02:00:37	38GB

Table 3.2: Indexing performance of Anserini, Indri and Terrier on smaller collections using 16 threads on a modest commodity server.

			Anserini (doc)		Indri		Terrier(single thread)	
Collection	docs	terms	time	size	time	size	time	size
Disk12	742k	219m	00:03:04	2.5GB	00:12:28	2.5GB	00:06:48	201m
Disk45	528k	175m	00:02:51	2.1GB	00:06:55	1.9GB	00:06:40	199m
AQUAINT	1.03m	318m	00:04:23	3.8GB	00:17:36	3.9GB	00:20:20	554m
WT2G	246k	182m	00:04:26	2.3GB	00:07:25	2.2GB	00:09:44	253m
WT10G	1.69m	752m	00:09:45	12GB	00:42:51	9.6GB	00:49:39	1.2GB
Gov2	25.2m	17.3b	06:40:15	331GB	14:51:12	215GB	20:41:49	17GB

The biggest issue, however, is that Lucene itself only provides access to a collection of indexing components that researchers need to assemble together to build an end-to-end indexer. For example, the developer would need to write from scratch custom document processing pipelines, code for managing individual indexing threads, and implementations of load balancing and synchronization procedures.

We address these issues in Anserini by providing abstractions for document collections that an IR researcher would be comfortable with, as well as the implementation of an efficient, high-throughput, multi-threaded indexer that takes advantage of these abstractions. Anserini models collections as comprised of individual segments (for example, the ClueWeb12 collection is comprised of a number of compressed WARC files) and provides implementations for common document formats—for parsing TREC-style XML documents, web pages stored in WARCs, tweets in JSON format, etc. In fact, Anserini ships with the ability to index many TREC collections "right out of the box". This greatly reduces the learning curve for researchers to get started with Lucene.

Table 3.3: Indexing performance of Anserini on web collections using 88 threads on a high-end server.

			Anserini (count)		Anserini (pos)	
Collection	docs	terms	time	size	time	size
CW09b	50m	31b	00:41	28GB	01:12	75GB
CW09	504m	268b	07:38	254GB	12:19	649GB
CW12b13	52m	31b	01:01	29GB	01:28	76GB
CW12	732m	429b	17:14	376GB	22:12	1.1TB

Streamlined IR evaluation (*extension*). Test collections play an important role in information retrieval research, and a substantial amount of research activity in improving ranking models is focused around *ad hoc* retrieval runs. A research toolkit should make this "inner loop" of IR research as easy as possible. Since Lucene was not originally designed for researchers, support for running experiments on standard test collections is largely missing. Anserini fills in this gap by implementing missing features: parsers for different query formats, a unified driver program for *ad hoc* experiments that outputs standard `trec_eval` format, etc. For convenience, existing TREC topics and qrels are included directly in our code repository—once again, reducing the learning curve for researchers to get started with Lucene.

There are two main uses for this feature in Anserini: First, our toolkit provides an easy way for researchers to replicate baselines of standard retrieval models such as BM25 and query likelihood. Armstrong et al. [3] previously identified the prevalent problem of weak baselines in experimental IR papers. Lin et al. [59] further noted that authors are often vague about the baseline parameter settings and the implementations they use. For example, Mühleisen et al. [75] reported large differences in effectiveness across four systems that all purport to implement BM25. Trotman et al. [97] pointed out that BM25 and query likelihood with Dirichlet priors can actually refer to at least half a dozen variants, and in some cases, differences in effectiveness are statistically significant. There is substantial community interest in engaging with reproducibility-related issues [2,35], and Anserini contributes to this discussion. Our proposed solution is to have widely-available baselines that are both competitive in effectiveness and easy to replicate. It is our hope that Anserini can fill this role.

Table 3.4: List of TREC topic and judgment files collected in Anserini.

Collection	Topics & Judgments
Disk12	51-200
Disk45	301-450 601-750
AQUAINT	ROBUST04 Hard
WT2G	401-450
WT10G	451-550
Gov2	701-850
CW09	51-200
CW12	201-300

Second, an easy-to-use baseline retrieval component in Anserini provides the starting point for additional ranking extensions. In particular, we advocate a multi-stage ranking architecture [4, 20, 81, 102] so that researchers will not need to directly work with native Lucene scoring APIs. That is, researchers should take advantage of Anserini APIs that generate an initial document ranking and hooks for feature extraction to build subsequent reranking stages. This, in fact, is the common architecture used in commercial web search engines today to support learning to rank [81].

Table 3.4 lists all topic and judgment files we have collected from TREC web site in order to ease the evaluation process.

We show some of the key commands to make the TREC evaluation as simple and easy as possible here. Anserini is open sourced and is hosted on Github. It is easy to get the code base by simply putting

```
git clone git@github.com:castorini/Anserini.git
```

After get the source code one can compile and build the binaries using

```
mvn clean package appassembler:assemble
```

We use Gov2 as the example to show how easy for the users of Anserini to streamline the TREC evaluation. Index the collection will be one command:

```
nohup sh target/appassembler/bin/IndexCollection
-collection Gov2Collection \
-input /path/to/gov2/ \
```

```
-generator JsoupGenerator \
-index lucene-index.gov2.pos+docvectors \
-threads 16 -storePositions \
-storeDocvectors -optimize \
 > log.gov2.pos+docvectors &
```

The directory /path/to/gov2/ should be the root directory of Gov2 collection, i.e., ls /path/to/gov2/ should bring up a bunch of subdirectories, GX000 to GX272. The command above builds a standard positional index (-storePositions) that's optimized into a single segment (-optimize). If one also wants to store document vectors (e.g., for query expansion), add the -docvectors option. The above command builds an index that stores term positions (-storePositions) as well as doc vectors for relevance feedback (-storeDocvectors), and -optimize force merges all index segment into one. After indexing is done, one should be able to perform a retrieval as follows:

```
sh target/appassembler/bin/SearchWebCollection \
-topicreader Trec \
-index lucene-index.gov2.pos+docvectors \
-bm25 \
-topics src/main/resources/topics-and-qrels/topics.701-750.txt \
-output run.gov2.701-750.bm25.txt
```

For the retrieval model: specify -bm25 to use BM25, -ql to use query likelihood, and add -rm3 to invoke the RM3 relevance feedback model (requires docvectors index). Topics and qrels are stored in src/main/resources/topics-and-qrels/. Use trec_eval to compute AP and P30:

```
eval/trec_eval.9.0/trec_eval \
src/main/resources/topics-and-qrels/qrels.701-750.txt \
run.gov2.701-750.bm25.txt
```

one should be able to get the same results shown in Table 3.6.

Relevance feedback (*extension*). Relevance feedback techniques provide robust solutions to the vocabulary mismatch problem between expressions of user information needs and relevant documents. Anserini provides a reference implementation of the RM3 variant of relevance models [54], built as a reranking module in the multi-stage architecture described above. Thus, our implementation is useful not only as a baseline for comparing query expansion techniques, but provides an example of how reranking extensions can be implemented in Anserini.

3.2.4 Evaluation

We describe experiments to support three claims about Anserini and the use of Lucene for information retrieval research. First, that Anserini is highly scalable and able to efficiently index large web collections. Second, that Anserini is similarly efficient in searching these collections and ranking documents using standard baseline models. Finally, Anserini is able to achieve scalable indexing and efficient retrieval without

The indexing performance of Anserini on a number of smaller and older collections is shown in Table 3.1 and Table 3.2. These experiments were conducted on a server with dual AMD Opteron 6128 processors (2.0GHz, 8 cores) with 40GB RAM running CentOS 6.8. This machine can be characterized as an old, modest commodity server. All experiments were run on an otherwise idle machine. With Anserini, we used 16 threads for indexing. and we report results from three different index configurations: count indexes where only term frequency information is stored (count), positional indexes that also store term positions (pos), and positional indexes that also store the raw documents and parsed document vectors (doc). In Table 3.1 we report the results for two different index configurations: count indexes where only term frequency information is stored (count), positional indexes that also store term positions (pos). For each condition, we report the indexing time in HH:MM:SS as well as the index size. The size of each collection is also shown for reference. We can see that Anserini can greatly reduce the size of the index and make the indexing fast.

In Table 3.2 we report another index configuration for Anserini: positional indexes that also store the raw documents and parsed document vectors (doc). As a comparison condition, we indexed the same collections using Indri 5.9 and Terrier 4.2 on the same machine. Indri enables the multi-threaded indexing (user can not explicitly set the number of threads to be used though) and its default indexing option is to include the term counts, positions, document vectors and the raw documents. So this can be seen as a fair comparison between Anserini and Indri. For Terrier, efficiency is one of its concerns so it should be included in the comparison. However, we have encountered difficulties when applying the multi-threaded option of it. So the indexing time of Terrier is using a single thread. Also, the index built with Terrier does not contain the raw documents. This may not be the fair comparison with the other two tools but we want to show here as the reference.

In Table 3.3, we report indexing performance for larger web collections on a server with dual Intel Xeon E5-2699 v4 processors (2.2GHz, 22 cores) and 1 TB RAM running Ubuntu 16.04. The table rows indicate different collections: CW09b refers to the ClueWeb09 (category B) web crawl, CW09 refers to all English pages in the ClueWeb09 web crawl, CW12b13 refers to the smaller ClueWeb12-B13 web crawl, and CW12 refers to the complete ClueWeb12 web crawl. Due to the size of the collections, we only report the count and positional index configurations. For these experiments, we used 88 threads for indexing on an otherwise idle machine; indexing time is reported in HH:MM. On this modern server, we are able to index *all* of ClueWeb12, one of the largest collections available to researchers today, in less than a day! As seen from Table 3.2, even on an older server, the indexing performance of Lucene is impressive. Compared to academic toolkits, Lucene does not appear to have any trouble scaling to large modern web collections.

Our next set of experiments were conducted on the Gov2 collection with Terabyte 06 efficiency queries. We issued all 100,000 queries sequentially against Indri, Terrier and Anserini indexes on the slower AMD Opteron server. Results are shown in Table 3.5, which reports latency (ms) and throughput (queries per second, or qps).

Table 3.5: Retrieval efficiency for Terabyte 06 efficiency queries on Gov2, using a single thread.

	Latency (ms)	Throughput (qps)
Indri	2403	0.42
Terrier	1358	0.72
Anserini	382	2.61

In this experiment, we used only a single query thread, and therefore do not take advantage of Lucene's ability to execute queries in parallel on multiple threads (so in our case, throughput is simply the inverse of latency). We see from these experiments that Lucene is roughly six times faster than Indri and is 3.5 times faster than Terrier.

Finally, we compared the retrieval effectiveness of Anserini and Indri. For Indri we refer to the RISE work of Yang and Fang [111], as they fine-tuned model parameters to achieve optimal effectiveness. We considered two baseline ranking models: Okapi BM25 (BM25) and query likelihood with Dirichlet priors (LM). For Anserini, we removed stopwords (the default) and tuned parameters as follows: for BM25 $k = 0.9$ and $b \in [0, 1]$ in increments of 0.1; for LM $u \in [0, 5000]$ in increments of 500. Results on standard TREC collections and queries are shown in Table 3.6, where (I) refers to Indri and (A) refers to Anserini. We see that effectiveness results are comparable between the two systems.

In summary, our experiments show that Anserini is at least as good as Indri in terms of effectiveness, and much faster in both indexing and retrieval. These results are consistent with findings from the recent Open-Source IR Reproducibility Challenge [59]. Together, empirical evidence presents a compelling case for adopting Lucene for information retrieval research.

3.3 Summary and Future Work

In this chapter we have discussed the tools for IR teaching. Specifically, we introduce VIRLab and Anserini as our efforts to push forward this matter. While VIRLab emphasizes the easy usage for the students or the beginners of IR, Anserini

Table 3.6: Effectiveness comparisons between Anserini and Indri on standard TREC test collections.

Collection	Disk12	Disk45	WT2G	WT10G	Gov2
Queries	51-200	301-450 601-700	401-450	451-550	701-850
BM25 (I)	0.2040	0.2478	0.3152	0.1955	0.2970
BM25 (A)	0.2267	0.2500	0.3015	0.1981	0.3030
LM (I)	0.2269	0.2516	0.3116	0.1915	0.2995
LM (A)	0.2232	0.2465	0.2922	0.2015	0.2951

focuses more on the scalability and the command line interface to deal with the large data collection on the single machine.

For the future work there are many interesting and challenging directions. First, to control what can be exposed to the users from the results due to data privacy concerns is a promising direction for web based teaching toolkits such as VIRLab. Second, it is nice to incorporate more state-of-the-art functionalities to Anserini, e.g. deep learning models for information retrieval as the baseline for this active field, which makes it more appealing.

Chapter 4

UNIFIED REPRODUCIBILITY EVALUATION SYSTEMS

In this chapter, we propose and develop several unified IR evaluation systems that aim to (1) solve the data privacy concern when doing the evaluation, (2) improve the reproducibility of IR research and (3) facilitate the evaluation and comparison of retrieval functions.

To address the data privacy problem, we propose to bridge the gap between academic research and industry data sets through a privacy-preserving evaluation platform. The novelty of the platform lies in its "data-centric" mechanism, where the data sit on a secure server and IR algorithms to be evaluated would be uploaded to the server. The platform will run the codes of the algorithms and return the evaluation results. Preliminary experiments with retrieval models reveal interesting new observations and insights about state of the art retrieval models, demonstrating the value of an industry data set.

For reproducibility study, a web based IR evaluation system RISE is proposed. With the help of RISE, more than 20 state of the art retrieval functions have been implemented and systematically evaluated over 16 standard TREC collections (including the newly released ClueWeb datasets). Our reproducibility study leads to several interesting observations. First, the performance difference between the reproduced results and those reported in the original papers is small for most retrieval functions. Second, the optimal performance of a few representative retrieval functions is still comparable over the new TREC ClueWeb collections. Finally, RISE is made publicly available so that any IR researchers would be able to utilize it to evaluate other retrieval functions.

4.1 Privacy Preserving Evaluation Platform (PPE)

4.1.1 Introduction

Evaluation is essential in the field of Information Retrieval (IR). Whenever a new IR technique is proposed and developed, it needs to be evaluated and analyzed using multiple representative data collections. Since the beginning of the field, there have been a few community-based efforts on constructing evaluation collections for the IR research, such as TREC, NTCIR and CLEF. These collections are available for researchers to download, and the researchers can then conduct experiments on these data collections using their own computers. Such an evaluation methodology has been used by many researchers in thousands of publications.

Although TREC collections can provide valuable insights on how well an IR method performs, they are not the same data collections used by the search engine industry. Unfortunately, privacy is one of the reasons that prevent industry from sharing their data sets [91]. As a result, it remains unclear how well the observations we draw about an IR method based on the TREC collections can be generalized to the real world data sets used in the search engine industry.

One possible solution is to anonymize the data to protect privacy [19]. However, the data anonymization would lead to the loss of some useful information, and it would also pose constraints on the developed methods. Recently, a data-centric evaluation methodology has been proposed [32, 34, 111]. This evaluation methodology does not require the sharing of the industry data set, which protects the privacy of the data. Instead, it advocates the industry to host an online evaluation system so that the researchers could upload their codes to evaluate their effectiveness over the industry data sets. A number of parallel works have been done under the name of Evaluation-as-a-Service (EaaS) [44, 60, 78] where the users of such system can leverage the APIs provided by the system to fetch documents or to submit a ranking request.

This paper follows the idea of the privacy-preserving evaluation (PPE) framework, and presents a specific implementation of the framework, i.e., the *PPE-M* system. With the implemented system, we evaluate a few representative basic IR models over an

industry data set and compare the results with those obtained on the standard TREC collections. Our study demonstrates that the PPE framework enables researchers to evaluate their methods using industry data collections, which essentially closes the gap between the IR researchers in academia and the industry data. Moreover, evaluation over the industry data set makes it possible to gain new insights about existing retrieval models. We focus on the evaluation of basic IR models in this paper, but the framework can be easily generalized for other tasks.

4.1.2 A General Framework of Privacy-Preserving Evaluation

Traditional IR evaluation methodology often requires a data collection to be downloaded to a local computer that also stores the code of an IR algorithm. After downloading the data, we can then run the code, get the results on the data collection and conduct further analysis. Clearly, this methodology would not work well for industry data sets which are not publicly available.

To address this limitation, a data-centric based privacy- preserving evaluation framework has been recently proposed [34]. The basic idea is to keep a data collection securely stored on its own server while allowing researchers to upload their codes to the server. The codes can access the statistics of the data collection through some pre-defined strategies, and then will be executed on the host server. The results will be returned to the researchers for further analysis.

There could be many different ways to implement such a general framework. In particular, we propose 3 different levels of support for evaluation that can accommodate different tradeoffs. The main idea is illustrated in Figure 4.1. The top level requires most work from the researcher's side but is most general as it can support evaluation of any algorithm in any language. The middle level provides API support, so the researcher can focus on implementing just the key component of the algorithm to be evaluated, but it requires the researcher to use a particular API. The low level provides an interactive evaluation Web interface and attempts to minimize a researcher's work, but the algorithm that can be supported in this way may also be limited by the code

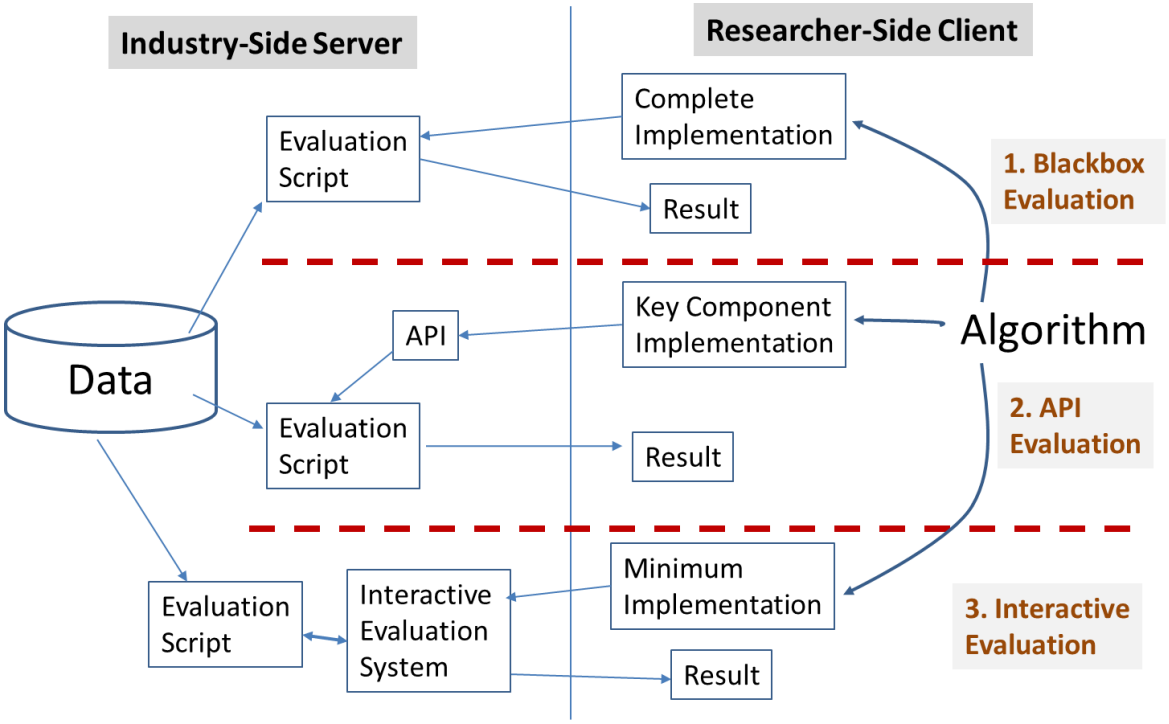


Figure 4.1: Three-level Support for PPE

that can be "opened up" by the system. An interactive system won't be able to provide full API support, so this would limit the algorithms that can be implemented and evaluated. It is clear that the top level is most general to support any algorithm, while the low level is most advanced with minimum effort on users, but has restriction on the algorithms to be evaluated.

We have already implemented the top level and the middle level, and will try to implement the low level in the future. Since the top level is trivial to implement, we focus on explaining how to implement the middle level in the next section.

4.1.3 A Specific Implementation

We now describe our implementation of the previously described privacy-preserving evaluation (PPE) framework. The implemented system focuses only on the evaluation of basic IR models, and is referred to as *PPE-M*.

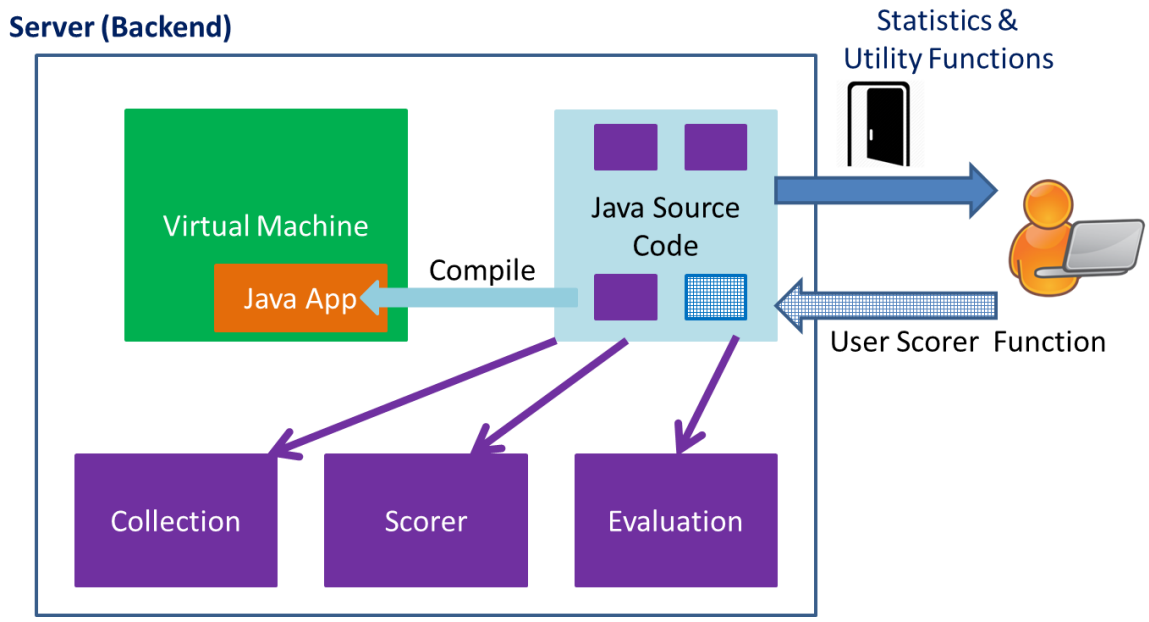


Figure 4.2: System Architecture

PPE-M is a web service with a typical client/server architecture. It hosts data collections on the server, and enables users to implement and submit their codes of retrieval models. Figure 4.7 shows the architecture of the implemented system. Once a code is uploaded to the server, it will be executed to retrieve documents from the collections. The retrieval results will be evaluated at the back end, the evaluation results based on standard measures such as MAP will be returned to the user for each data collection. The system is available at http://xxx.yy.z (anonymized for blind reviewing).

The front end of the system is a web form, which allows the users to upload a Java source code file that includes the implementation of a retrieval function. The screenshot of the code submission interface is shown in Figure 4.3. Users can first select which task to participate, and then upload the source code as well as the data file (if necessary). The task could be ad hoc retrieval task, recency-based retrieval task, etc. The source code includes the implementation of a retrieval function in Java. The data file is optional, and it could include some prior information. Since the code will be executed on the server, it has to follow some conventions on accessing the

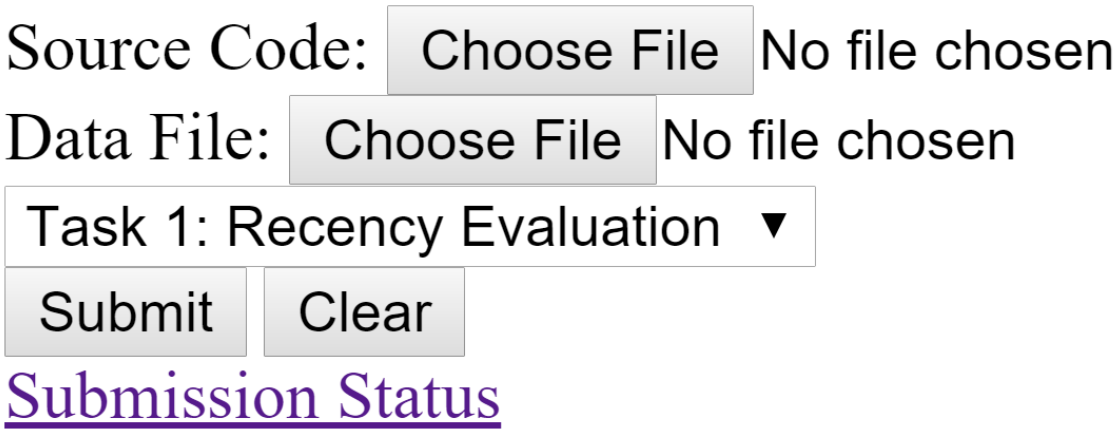


Figure 4.3: Screenshot of code submission interface

Code Id	Task Id	Data Attached	Submitted Time	Status	Message
2538	0	false	2016-02-10 14:27:32	FAILED	Compilation Error.
2537	0	false	2016-02-10 14:27:32	FINISHED	ndcg3:0.7145 map:0.8397
2536	0	false	2016-02-10 14:17:49	FINISHED	ndcg3:0.7145 map:0.8397
2535	0	false	2016-02-10 14:17:49	FINISHED	Exception in thread "main" java.lang.NullPointerException

Figure 4.4: Screenshot of the result page

collection statistics and calling external functions. To help users get familiar with these conventions, a user guide and example codes are provided. The user guide includes a list of currently supported collection statistics that can be accessed by the code as well as a list of utility functions that the code can call. The restrictions posed on the codes are to prevent potential malicious attack from outsiders while making it possible for users to leverage the provided statistics to evaluate their models.

The core component of the *PPE-M* system is the Java source code package. It consists of several modules, and each of them is responsible for a functionality. The *code compilation module* takes the uploaded code of retrieval, compile it and execute it on the virtual machines. After the code execution, the system computes the relevance

scores of documents and generates the ranking list for each document. This process is done by the *result generation* module. The *collection process* module is used to pre-process documents in the collection, and the pre-processing could be tokenization, stop words removal, stemming, etc. The *evaluation* module takes the ranking list and then generate the evaluation results based on various measures. Such a modularized architecture offers flexibility in the implementation of the framework since each module could be re-implemented and tested independently.

The system returns the evaluation results to users through an interface as shown in Figure 4.4. If the code can be compiled and executed correctly, the evaluation results will be returned, as shown in the last column. Otherwise, error messages will be displayed. Users are able to see how well their ranking models perform over each available data set. Moreover, they are able to see the evaluation results of codes submitted by others. In the future, we plan to further enhance the interface with a leader-board that sort and display all the submitted runs based on their performance.

The *PPE-M* system is implemented and designed in the above way for the following reasons. First, the system preserves the privacy of the industry data collections. The data collections are not distributed to users but are stored on the server. Users' code may only access certain type of information about the collection and use them to compute the relevance scores, but the collection information would not be passed to the client side. Second, the system is configurable based on the level of the privacy concerns about the data collections. For example, if more information can be released about a data collection, users can use the information in their codes or access more information from the evaluation results. Finally, the system can be easily generalized to evaluate other tasks in IR such as recency-based retrieval and click-prediction.

A new industry data set is available for evaluation through *PPE-M*. This data set has been routinely used in an industry lab. The data set contains 3,274 news-related queries and 71,406 articles. The queries were collected across a few months. For each query, around 20 documents are selected from all the news articles based on the ranking produced by a very simple retrieval method. For each query, editors manually assign

Table 4.1: Statistics of Test Collections

Collections	IC	ROBUST04	WT2G	GOV2
#queries	3,274	250	50	150
avg(ql)	2.80	2.73	2.44	3.11
avg(idf(qt))	13.75	11.50	9.81	13.49
#documents	71,406	528,155	247,491	25,205,179
avg(dl)	583.44	467.55	1057.59	937.25

each document with a relevance label (1-Bad, 2-Fair, 3-Good, 4-Excellent).

4.1.4 Experiments

We conduct preliminary experiments using the data set provided in *PPE-M* system to verify observations about basic retrieval models that people have made previously on TREC data sets. As the results will show, the new data set is useful since it can provide new insights on existing retrieval models.

4.1.4.1 Experiment Design

We denote the industry data set described earlier as *IC*. In addition to this data set, we also report results on a few representative TREC collections: *ROBUST04*, *WT2G* and *GOV2*. These data sets are selected to cover different types and sizes of the collections. The statistics of all the collections are summarized in Table 5.2.

We compare three representative retrieval functions: (1) Okapi BM25 (**BM25**) [87]: a function derived from the classic probabilistic model; (2) Pivoted document length normalization (**Piv**) [92]: a function derived from the vector space model; and (3) F2EXP (**F2EXP**) [33]: a function derived using axiomatic approaches. These three functions are selected because they are among the most effective retrieval functions based on the evaluation over multiple TREC collections. *F2EXP*, in particular, has been shown to be more robust than existing retrieval functions with comparable optimal performance. The main difference between *F2EXP* and other retrieval functions lies in its different implementation of IDF and document length normalization parts.

Table 4.2: Optimal Performance Comparison (MAP). Optimal parameter settings are reported in parenthesis.

Model	IC	ROBUST04	WT2G	GOV2
BM25	0.8687 (0.35)	0.2478 (0.20)	0.3152 (0.15)	0.2970 (0.35)
PIV	0.8693 (0.20)	0.2206 (0.05)	0.2945 (0.05)	0.2536 (0.05)
F2EXP	0.8595 (0.00)	0.2512 (0.30)	0.2973 (0.25)	0.2828 (0.25)

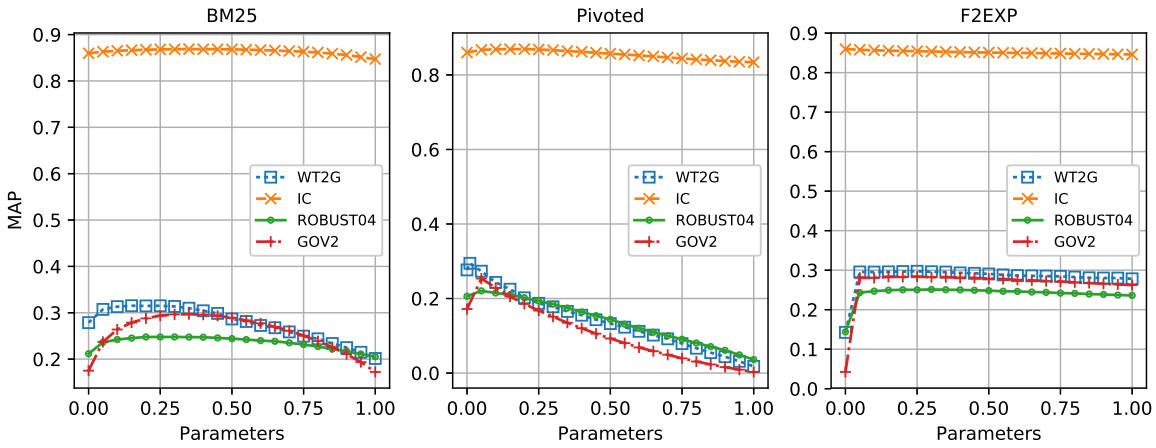


Figure 4.5: Parameter Sensitivity (MAP)

4.1.4.2 Retrieval Performance Comparison

We compare the optimal performance of the retrieval functions over all the data sets and summarize the results in Table 4.2. Figure 4.5 shows the parameter sensitivity curves.

In general, the results on the *IC* data set are consistent with those on the TREC collections. Specifically, the optimal performance of the three functions are comparable. The optimal parameters are also within the reasonable range as mentioned in the previous study [30]. However, there are also a few new interesting observations that we can make based on the results from the industry data set.

The first interesting observation is that the evaluation results on the *IC* data set are much higher than on the TREC collections. This is not surprising since the

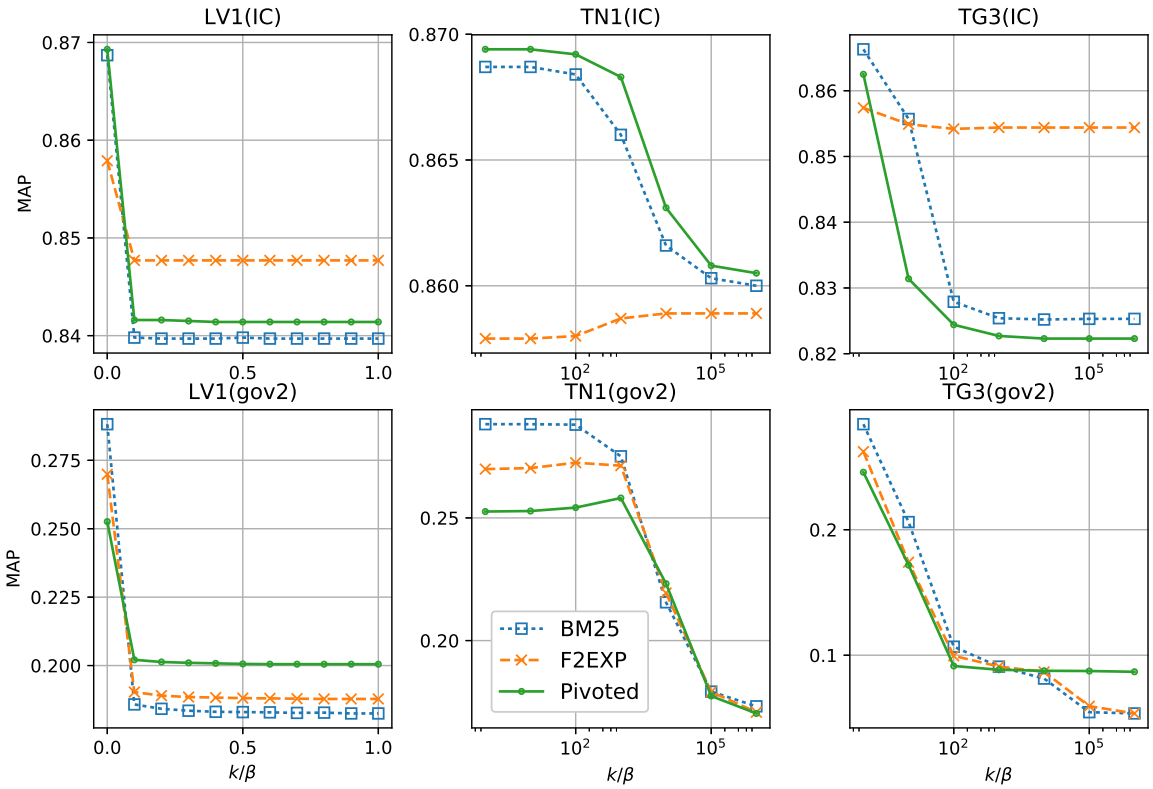


Figure 4.6: Results of perturbation tests

IC data set is constructed by pooling top ranked documents for each query based on a simple ranking method while the documents of TREC collections are selected independently to the queries. Since most Web search engines now adapt a multi-level ranking strategy [101], the *IC* data set actually represents a more realistic problem setup.

The second interesting observation is about the *F2EXP* function. Although *F2EXP* has been shown to be robust in terms of the parameter values, its optimal parameter value is always larger than 0. However, its optimal parameter value is equal to 0 for the *IC* data set, which indicates that its length normalization part is not very effective. This is something that we have never observed based on the results for TREC collections.

4.1.4.3 Further Analysis

So far, we have demonstrated that the *PPE-M* system is able to evaluate retrieval models without releasing the data set. One new discovery made using this data set is about the "unusual" optimal parameter value in the *F2EXP* function.

To look into the reason behind this observation, we conduct more analysis using diagnostic evaluations [30]. The diagnostic evaluation methodology was proposed to identify the weaknesses and strengths of retrieval functions based on the perturbation of collections [30]. Each perturbation is designed to test a specific aspect of a retrieval function. Some perturbations can be done by changing simple statistics, while others may require additional information about the collections. In this paper, we only apply perturbation tests that can be implemented using the available statistics provided by the *PPE-M* system. These tests include two length variance sensitivity tests, one term noise resistance test and three TF-LN balance tests.

Figure 4.6 shows the perturbation results. The plots on the first row are for the *IC* data set, and those on the second row are for the *GOV2* data set. Due to the space limit, we only show the results of the tests that are different on the two sets, so that we can focus on new insights gained by using the industry data set.

The first perturbation test is the length variance reduction test (LV1). We prefer curves that are lower because they indicate the functions would have more gain on length normalization part. Two plots on the first column in Figure 4.6 indicate that *F2-EXP* has less gain on length normalization part for the *IC* data set, which is something we fail to observe from the TREC data set.

The second test is the term noise resistance test (TN). The curves that are higher means that they penalize long documents more appropriately. The plots on the second column in Figure 4.6 suggest that *F2-EXP* did a poor job to penalize long documents with more noisy terms on the *IC* data set. However, such a trend is not clear based on the results from the TREC data set.

The third test is the all query term growth test (TG3). We prefer curves that are higher since it means the corresponding function can balance TF and LN more

appropriately. The last column in Figure 4.6 indicates that *F2-EXP* did a better job to avoid over-penalize long documents with more query terms on the *IC* data set. And this is something we can not see based on the results on the TREC data set.

In summary, our preliminary study has demonstrated the possibility of using the implemented *PPE-M* to evaluate IR models using a real world industry data set. More interestingly, we are able to gain new insights about existing retrieval functions by using the industry data.

4.2 RISE - A Reproducibility Platform for Retrieval Models

To reproduce the results of retrieval models, we implement a web-based Reproducible Information retrieval System Evaluation (*RISE*) platform. The platform is designed to provide a well-controlled environment for the users to implement and evaluate retrieval functions. Figure 4.7 shows the architecture of RISE. *RISE* is basically a web service built on top of a modified version of the Indri[1] toolkit. *RISE* hosts data collections on the server side, processes documents, and builds the indexes. Users need to upload their own implementations of retrieval functions based on the provided templates. After the code is uploaded, *RISE* automatically compiles it and evaluates it over the selected data collections. The evaluation results of the retrieval function will then be added to the score boards and thus be available for comparison.

Any registered users can contribute the implementation of a retrieval model to the system. Users are expected to be familiar with C++ but not necessarily familiar with Indri, as we provide detailed instructions with sample codes on how to access the statistics from the indexes and how to implement ranking models with the provided statistics. Moreover, *RISE* is an open system which allows any user to view any other users' implementations of the models. This functionality makes it possible for a user to easily try different variants of existing retrieval functions. The modified version of the Indri toolkit provides various statistics that are not available in the original version.

[1] http://sourceforge.net/projects/lemur/

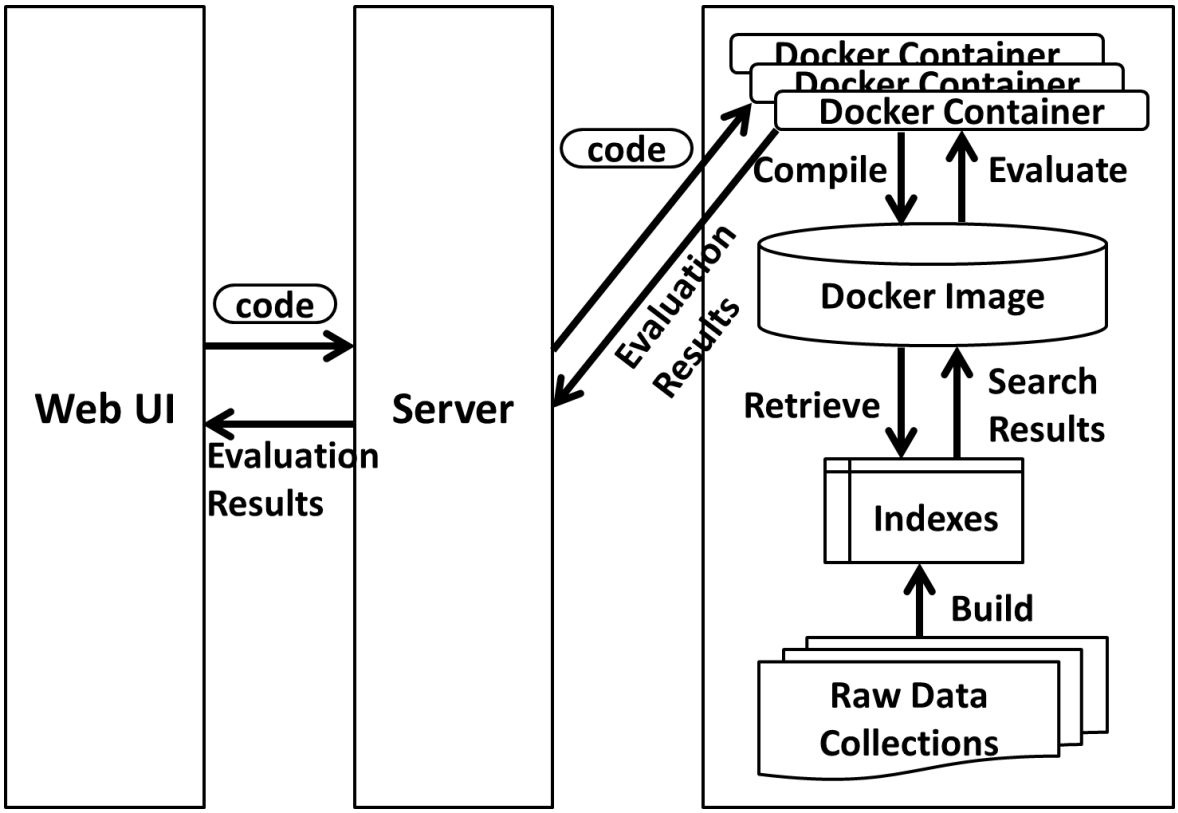


Figure 4.7: System Architecture

These new statistics include query term frequency, average document term frequency, etc.

After the code is submitted to the server and successfully compiled, a Docker container is temporarily initiated on top of the static Docker image. (Docker container is like a sandbox which provides an isolated environment acting like an operating system. For more information please refer to https://www.docker.com/) The Docker image includes the indexes built from data collections and the modified Indri toolkit that will be used as the facility to run the model and generate the ranking list. A Docker image can be utilized by several Docker containers at the same time while keep the same underlying view of index and thus is the ideal choice for the system. Several Docker containers can be initiated in parallel so that multiple models can be compiled, run and evaluated at the same time. Moreover, by carefully setting the Docker container we can control the CPU and memory usage as well as security related

settings (e.g. network, 3rd party libraries) so that the system is more robust against malicious/careless usage. The running Docker container compiles the codes, generate the ranking list, then evaluate the results. After that, the performance is rendered to the users and users can opt to choose other models to compare with.

There are several major benefits of the developed *RISE* platform. (1) The data collections are kept on the server side to preserve the data privacy. (2) The platform uses Docker to control the evaluation environment, which is more secure, faster and more reliable. (3) The platform provides a repository of the implementation of various retrieval functions. Registered users can upload their own codes, and these codes can be reused by other users. Such a repository could eliminate redundant efforts of implementing baseline methods among IR researchers. (4) The platform maintains the scores for each implemented retrieval function over all available data sets. The score boards could become a valuable reference when a new retrieval function needs to be evaluated and compared with the state of the art methods.

4.2.1 Reproduced Retrieval Functions

With the developed *RISE* platform, we conduct a reproducibility study of IR models with 21 representative retrieval functions. These retrieval functions include the representative ones from the vector space models [77,92], the classic probabilistic models [87], the language modeling approaches [117,118], the divergence from randomness models [1], the axiomatic models [33,69], and the information theory based models [21].

Let us first explain the notations used in the paper.

- $|q|$: the number of terms in query q
- c_t^q: the number of occurrences of term t in query q
- c_t^d: the number of occurrences of term t in document d
- l_d: the length of document l
- c_d: the number of unique terms in document d
- F_t: the total number of term t in collection
- N_t: the number of documents containing term t

Table 4.3: Retrieval functions that are reproduced in our study (Part 1)

Okapi BM25 and its variants	
BM25	$\sum_{t \in q} \frac{(k_3+1) \cdot c_t^q}{k_3 + c_t^q} \cdot \frac{(k_1+1) \cdot c_t^d}{c_t^d + k_1 \cdot (1 - b + b \cdot \frac{l_d}{L})} \cdot ln\left(\frac{N - N_t + 0.5}{N_t + 0.5}\right)$
F2EXP	$\sum_{t \in q} \frac{c_t^d}{c_t^d + s + s \cdot \frac{l_d}{L}} \cdot \left(\frac{N+1}{N_t}\right)^k$
F2LOG	$\sum_{t \in q} \frac{c_t^d}{c_t^d + s + s \cdot \frac{l_d}{L}} \cdot ln\left(\frac{N+1}{N_t}\right)$
BM3	$\sum_{t \in q} \frac{(k_3+1) \cdot c_t^q}{k_3 + c_t^q} \cdot \frac{(k_1+1) \cdot tfn}{k_1 + tfn} \cdot ln\left(\frac{N - N_t + 0.5}{N_t + 0.5}\right)$ $tfn = \frac{c_t^d + \mu \cdot \frac{F_t}{\lvert C \rvert}}{l_d + \mu} \cdot \mu$
BM25+	$\sum_{t \in q} \frac{(k_3+1) \cdot c_t^q}{k_3 + c_t^q} \cdot \left[\frac{(k_1+1) \cdot c_t^d}{c_t^d + k_1 \cdot (1 - b + b \cdot \frac{l_d}{L})} + \delta\right] \cdot ln(\frac{N+1}{N_t})$
Pivoted and its variants	
PIV	$\sum_{t \in q} \frac{1 + ln(1 + ln(c_t^d))}{(1-s) + s \cdot \frac{l_d}{L}} \cdot ln\left(\frac{N+1}{N_t}\right)$
F1EXP	$\sum_{t \in q} (1 + ln(1 + ln(c_t^d))) \cdot \frac{L+s}{L + s \cdot l_d} \cdot \left(\frac{N+1}{N_t}\right)^k$
F1LOG	$\sum_{t \in q} (1 + ln(1 + ln(c_t^d))) \cdot \frac{L+s}{L + s \cdot l_d} \cdot ln\left(\frac{N+1}{N_t}\right)$
PIV+	$\sum_{t \in q} \left[\frac{1 + ln(1 + ln(c_t^d))}{(1-s) + s \cdot \frac{l_d}{L}} + \delta\right] \cdot ln\left(\frac{N+1}{N_t}\right)$
NTFIDF	$\sum_{t \in q} \left[\left[\omega \cdot f\left(\frac{c_t^d}{l_d / c_d}\right) + (1 - \omega) \cdot f\left(c_t^d \cdot log_2\left(1 + \frac{L}{l_d}\right)\right)\right] \cdot \left[ln\left(\frac{N+1}{N_t}\right) \cdot f\left(\frac{F_t}{N_t}\right)\right]\right]$ $\omega = \frac{2}{1 + log_2(1 + \lvert q \rvert)}$, $f(x) = \frac{x}{1+x}$

- $|C|$: the number of terms in collection
- N: the number of documents in collection
- L: the average document length in collection

We now provide more details about the retrieval functions that are included in the reproducibility study. All the implemented functions are summarized in Table 5.1 and Table 4.4.

4.2.1.1 Okapi BM25 and Its Variants

Okapi BM25 is one of the representative retrieval functions derived from the classical probabilistic retrieval model. It was first proposed at TREC-3 [87], and has become one of the most commonly used baseline retrieval functions. This function is denote as **BM25** in the paper.

Axiomatic approaches was first applied to Okapi BM25 to develop new retrieval functions [33] in 2005. The basic idea is to search for a retrieval function that can satisfy all reasonable retrieval constraints. Instead of blindly search for the function, one strategy is to start with an existing retrieval function, such as Okapi BM25, find its general form, and use the retrieval constraints to find different instantiations that can satisfy more retrieval constraints. The previous study derived two variants based on Okapi BM25, and they are referred to as **F2EXP** and **F2LOG** in the paper. Compared with the original BM25 function, these two variants have different implementations for both term frequency (TF) normalization part and the inverse document frequency (IDF) part.

Another variant of BM25 came from the study of Dirichlet Priors for term frequency normalization [43]. This variant, denoted as **BM3**, replaces the original TF normalization components in the BM25 function with the Dirichlet Priors TF normalization component.

Following the axiomatic methodology, Lv and Zhai [69] revealed a deficiency of the BM25 in its TF normalization component, i.e., the TF normalization component is not lower-bounded properly. To fix this problem, a variant of BM25, denoted as **BM25+**, was proposed. The main change is to add a lower bound to the TF normalization part.

4.2.1.2 Pivoted Normalization Function and Its Variants

Pivoted normalization method, denoted as **PIV**, is one of the most representative retrieval functions derived from the vector space model [92]. It can be regarded as one of the best-performing TF-IDF retrieval functions.

Axiomatic approaches were also applied to derive variants of the pivoted normalization method [33]. The two variants are denoted as **F1EXP** and **F1LOG**. Compared with the original function, the two variants are different in their implementations of IDF and TF normalization.

Similar to BM25, low-bounding term frequency normalization has also been applied to the pivoted function. The variant is denoted as **PIV+**, and it differs from the original function in having a lower bound added to the TF normalization component.

A novel TF-IDF term weighting scheme was proposed in 2013 to capture two different aspects of term saliency [77]. In particular, its TF component is a combination of two normalization strategies, in which one prefers short documents while the other prefers long documents. Its form is quite different from the pivoted normalization function. We include it as one of the variants for the pivoted normalization function because it uses a novel TF-IDF weighting strategy.

4.2.1.3 Language Modeling Approaches

Dirichlet prior method, denoted as **DIR**, is one of the representative retrieval functions derived using the language modeling approaches [118]. It uses the Dirichlet prior smoothing method to smooth a document language model and then ranks the documents based on the likelihood of the query is generated by the estimated document language models.

Two-stage language models were proposed to explicitly capture the difference influences of the query and document collection on the optimal parameter setting [117]. Compared with the Dirichlet prior method, the two-stage smoothing method (denoted as **TSL**) interpolates the smoothed document language model with a query background language model.

Instead of assuming document models take the form of a multinomial distribution over words, Multiple-Bernoulli language models assume that the document is a sample from a Multiple-Bernoulli distribution [73]. The retrieval function is denoted as **BLM**.

Similar to BM25 and Pivoted, Dirichlet prior method has also been studied using axiomatic approaches. Two variants derived using the axiomatic approaches [33] are denoted as **F3EXP** and **F3LOG**. The variant derived based on the lower bound term frequency normalization [69] is denoted as **DIR+**.

4.2.1.4 Divergence from Randomness Models

The **PL2** model is a representative retrieval function of the divergence from randomness framework [1]. It measures the randomness of terms using Poisson distribution with Laplacian smoothing.

The first variant of the PL2 is to replace the original TF normalization component with the Dirichlet prior TF normalization [43]. This variant is denoted as **PL3**.

The second variant of the PL2 considered in this paper is to apply the lower bound term frequency normalization [69]. It is denoted as **PL2+**.

4.2.1.5 Information-based Models

A family of information-based models was proposed for ad hoc IR [21]. These models focused on modeling relevance based on how a word deviates from its average behavior. Two power law distributions (e.g., a smoothed power-law distribution and log-logistic distribution) were used, and the corresponding functions are denoted as **SPL** and **LGD**.

4.2.2 Experiments

We now describe the experiment design and results for our reproducibility study. The first set of experiments mainly focuses on whether we can reproduce the retrieval results that have been reported in the previous studies and whether the reproduced results are consistent with that have been reported. The second set of experiments aims to examine how well the retrieval functions perform on the newly released data sets and checks whether the conclusions are consistent with the previous findings. Finally, we also provide reference performance for all the reproduced retrieval functions over a wide range of TREC collections including the newly released ClueWeb collections.

4.2.2.1 Reproducibility Study

4.2.2.2 Experiment Design

For the reproducibility experiments, we conduct experiments over 11 data sets that have been used in the ad hoc retrieval task at TREC-1, TREC-2, TREC-3, TREC-6, TREC-7, TREC-8; the small web track at TREC-8; the terabyte track at TREC 2004-2006; and the robust track at TREC 2004. The statistics of the data collections are summarized in Table 4.5.

All the collections are stemmed using Porter's stemmer. We mainly focus on the title part of the query topics. If the performance of title query is not reported by the original paper, then we use whatever query (e.g. description part or title+description+narrative) that was originally used. Please note that for some papers the authors reported the performances on the combination of multiple query topic sets, e.g. TREC678 as one query set. For this kind of query we treat the three years' topics as one query set like what the original authors did.

We evaluate the retrieval functions over these data collections and compare our results with what have been reported in the previous studies. The results are evaluated with MAP@1000, and the evaluation results are computed using `trec_eval`[2].

4.2.2.3 Results

We evaluate the retrieval performance for each of the 21 retrieval functions described in the previous section over all the data collections mentioned in Table 4.5. We then compare our reproduced results of a retrieval function with the original results reported in the paper that proposed the function. Due to the space limit, we can not report all the reproduced results, so we summarize a few main findings here.

WT2G and **disk4&5** are the two commonly used document collections in the previous study. We summarize the performance comparison between the reproduced results and the original results on these two data sets in Table 4.6 and Table 4.7

[2] http://trec.nist.gov/trec_eval/

respectively. Note that **disk4&5** refers to all the data sets that use disk 4 and 5 as document collections, and it includes TREC6, TREC7 and TREC8. Let us first explain the notations in the two tables. The `orig.` column lists the originally reported results. The `repd.` column are the reproduced results. Either positive or negative difference between `orig.` and `repd.` is shown as percentage w.r.t the `orig.` in column `diff.`. The free parameter(s) used by the original paper are reported in column `para.` where * means the parameter is not explicitly reported in the original paper and we just pick the optimal one by grid search. The original paper of BM25 and PIV did not report the performances on the collections that we select. Instead, we use what were reported in [43,69] for these two models as their `orig.` results. Note that some retrieval functions are missing from the table because their original papers did not report the performance on the corresponding collection.

The results show that the performance differences with respect to the original performance, i.e, `diff.`, are small. Most of them are in the range of $[-5\%, +5\%]$. This indicates that we are able to successfully reproduce the retrieval performance for these functions.

To gain a better understanding of the reproduced results for all retrieval function, we summarize the performance difference (both mean and standard deviation) between the original and reproduced results for each of the retrieval function. The results are shown in Table 4.8. Although the reproduced results are not exactly the same as what were reported, the differences are generally small. We do not have the results for BLM because the authors of that paper did not report the performances on any collection that we have selected.

Among all the retrieval functions, PL2 has the largest standard deviation for the performance differences, and NTFIDF has the largest mean performance difference. We provide more detailed reproduced results for these two functions in Table 4.9. It is clear that the performance differences are consistent over almost all the collections. One possible explanation is that these two functions were originally implemented using

the Terrier[3] retrieval system as opposed to Indri used in our paper. As pointed out in the previous study [74], using different toolkits could lead to different evaluation results.

4.2.2.4 Performance Comparison on Web Search Collections

Not only can the *RISE* system provide a platform to reproduce the results of existing IR models, but also minimize the efforts when evaluating IR models over new collections. Whenever there is a new data collection available, the *RISE* system can easily run all the implemented retrieval functions on the new data collection and generate evaluate results for each function.

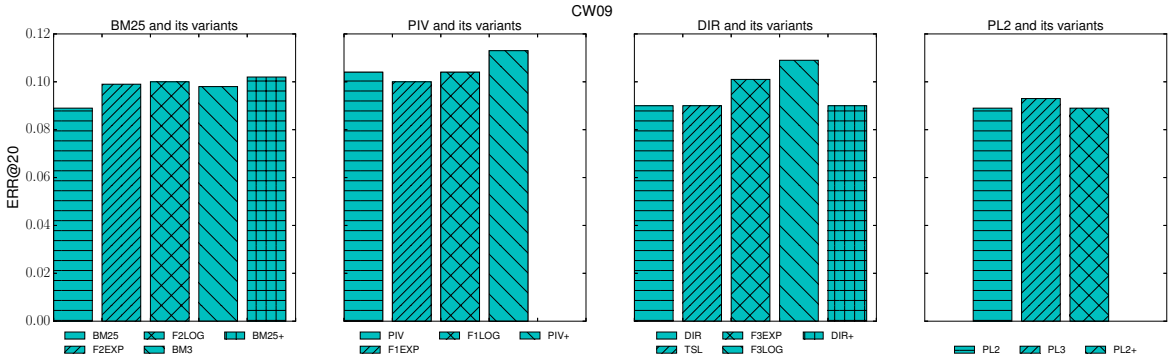
We conduct experiments to evaluate the performance of retrieval functions over 5 data sets used in the Web track from TREC 2010 to TREC 2014. The Web track at TREC 2010 to TREC 2012 used the ClueWeb09[4] as the document collection. Each year's Web track has 50 topics. Since the entire ClueWeb09 collection is too big to host on our server, we used the category B colleciton, which contains a subset of about 50 million English pages. The Web track at TREC 2013 to TREC 2014 used the ClueWeb12[5] as the document collection. Each data set has 50 topics developed by NIST. Again, due to the huge size of the original ClueWeb12 data set, we evaluate the retrieval functions over a subset of collection. The subset is generated by sampling documents from the raw collection. We use Indri default query likelihood baseline to retrieve top 10,000 documents for each query and make these documents as the sampled collection. Following the measured used at the TREC Web track, ERR@20 is used to evaluate the performance for these data sets. Due to the space limit, instead of reporting the performance over each Web track data set, we report the performance based on the document collection used. For example, **CW09** corresponds to the data

[3] http://terrier.org/

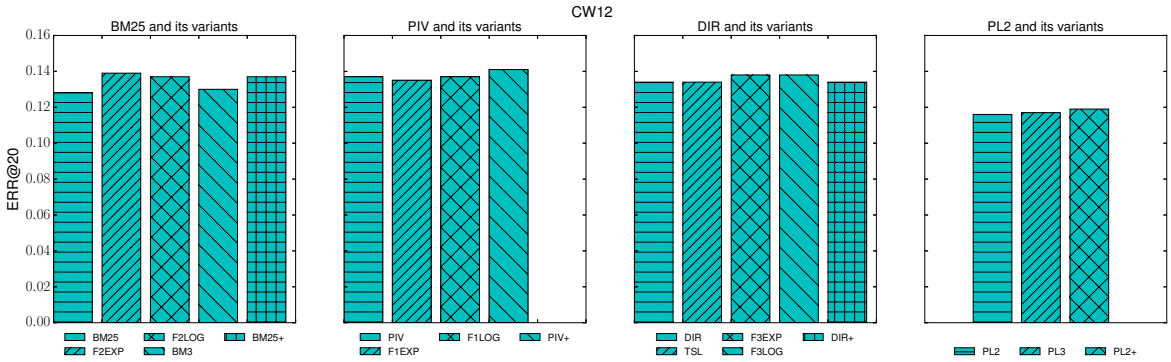
[4] http://lemurproject.org/clueweb09/

[5] http://lemurproject.org/clueweb12.php/

Figure 4.8: Optimal Performances on ClueWeb Collections



(a) Performances of selected models on CW09



(b) Performances of selected models on CW12

set combining data used in the Web track at TREC 2010-2012. Similarly, **CW12** corresponds to the data set combining data used in the Web track at TREC 2013-2014.

As discussed in the previous section, many variants have been proposed to improve the performance of representative retrieval functions such as BM25, PIV, DIR and PL2. All those studies were conducted over the traditional TREC collections. Thus, it would be interesting to see whether the improvement would still exist on the new Web collections.

Figure 4.8 shows the optimal performance comparison of the representative retrieval functions with their variants on the new Web collections. We can make a few interesting observations. First, it is interesting to see that most variants can outperform their original retrieval functions. For example, all the variants of BM25 performs

better than BM25 on both collections. The only exception is the PIV function. PIV performs really well on the two new collections. Second, divergence from randomness models do not perform as well as other retrieval functions. Finally, the optimal performances of the BM25 variants, PIV variants and DIR variants are comparable.

4.2.2.5 Summary

To serve as a future reference, we summarize the optimal performance of all the retrieval functions over all the data sets in Table 4.10. Due to the space limit, the data sets are categorized based on the collections used, so data sets used in multiple tracks might be grouped into one because they used the same document collections. For each retrieval function, the free parameters are tuned via grid search and the parameter ranges are summarized in Table 4.11.

The optimal performances for the selected retrieval models on all collections are shown in Table 4.10. To the best of our knowledge this is the first time of reporting such large scale and comprehensive performances of retrieval models.

4.3 Summary and Future Work

We have shown the level-based privacy preserved evaluation systems. The advantage of such systems include the erase of redistribution of private data collections and standard evaluation environment. We have described two proposed systems: PPE and RISE. The former is a standard practice for industrial data sets and the latter provides unified evaluation environment for the IR research community.

For future work, we think level 1 and level 3 implementation of the PPE systems should be promising. Level 1 introduces great challenge which needs the rethinking of the whole IR system – from indexing to evaluation. Level 3 implementation is obviously a good example to promote IR techniques to non-experts.

Table 4.4: Retrieval functions that are reproduced in our study (Part 2)

Language modeling approaches	
DIR	$\sum_{t \in q} ln\left(\frac{c_t^d + \mu \cdot \frac{F_t}{\lvert C \rvert}}{l_d + \mu}\right)$
TSL	$\sum_{t \in q} \left((1-\lambda) \cdot \frac{c_t^d + \mu \cdot \frac{F_t}{\lvert C \rvert}}{l_d + \mu} + \lambda \cdot \frac{F_t}{\lvert C \rvert}\right)$
BLM	$\sum_{t \in q} \frac{c_t^d + \mu \cdot \frac{F_t}{\lvert C \rvert}}{l_d + \frac{\lvert C \rvert}{F_t} + \mu - 2}$
F3EXP	$\sum_{t \in q} (1 + ln(1 + ln(c_t^d))) \cdot \left(\frac{N+1}{N_t}\right)^k - \frac{(l_d - \lvert q \rvert) \cdot \lvert q \rvert \cdot s}{L}$
F3LOG	$\sum_{t \in q} (1 + ln(1 + ln(c_t^d))) \cdot ln\left(\frac{N+1}{N_t}\right) - \frac{(l_d - \lvert q \rvert) \cdot \lvert q \rvert \cdot s}{L}$
DIR+	$\sum_{t \in q} \left[ln\left(1 + \frac{c_t^d}{\mu \cdot \frac{F_t}{\lvert C \rvert}}\right) + ln\left(1 + \frac{\delta}{\mu \cdot \frac{F_t}{\lvert C \rvert}}\right)\right] + \lvert q \rvert \cdot ln \frac{\mu}{l_d + \mu}$
Divergence from Randomness Models	
PL2	$\sum_{t \in q} \frac{tfn \cdot log_2(tfn \cdot \lambda) + log_2 e \cdot (\frac{1}{\lambda} - tfn) + 0.5 \cdot log_2(2\pi \cdot tfn)}{tfn+1}$ $tfn = c_t^d \cdot log_2\left(1 + c \cdot \frac{L}{l_d}\right)$
PL3	$\sum_{t \in q} \frac{tfn \cdot log_2(tfn \cdot \lambda) + log_2 e \cdot (\frac{1}{\lambda} - tfn) + 0.5 \cdot log_2(2\pi \cdot tfn)}{tfn+1}$ $tfn = \frac{c_t^d + \mu \cdot \frac{F_t}{\lvert C \rvert}}{l_d + \mu} \cdot \mu$
PL2+	$\sum_{t \in q, \lambda > 1} \left[\frac{tfn \cdot log_2(tfn \cdot \lambda) + log_2 e \cdot (\frac{1}{\lambda} - tfn) + 0.5 \cdot log_2(2\pi \cdot tfn)}{tfn+1} + \frac{\delta \cdot log_2(\delta \cdot \lambda) + log_2 e \cdot (\frac{1}{\lambda} - \delta) + \frac{log_2(2\pi\delta)}{2}}{\delta+1}\right]$ $tfn = c_t^d \cdot log_2\left(1 + c \cdot \frac{L}{l_d}\right)$
Information-based Models	
SPL	$\sum_{t \in q} -ln\left(\frac{\frac{n_t^d}{n_t^d+1} - \lambda_t}{1 - \lambda_t}\right)$ $\lambda_t = \frac{F_t}{N}$ and $n_t^d = c_t^d + ln\left(1 + c \cdot \frac{L}{l_d}\right)$
LGD	$\sum_{t \in q} -ln\left(\frac{\lambda_t}{n_t^d + \lambda_t}\right)$ λ_t and n_t^d as shown above

Table 4.5: Data collections used for the reproducibility study

	Topics	Doc. collection	#documents	avdl
ad hoc task at TREC-1	51-100			
ad hoc task at TREC-2	101-150	**disk1&2**	741,856	412.89
ad hoc task at TREC-3	151-200			
ad hoc task at TREC-6	301-350			
ad hoc task at TREC-7	351-400	**disk4&5**	528,155	467.553
ad hoc task at TREC-8	401-450			
robust track at TREC 2004	601-700			
small web task at TREC-8	401-450	**WT2G**	247,491	1057.59
terabyte track at TREC 2004	701-750			
terabyte track at TREC 2005	751-800	**GOV2**	25,205,179	937.252
terabyte track at TREC 2006	801-850			

Table 4.6: Performance comparison of reproduced and original results on **WT2G**

Models	**orig.**	**repd.**	**diff.**	**para.**
BM25 and its variants				
BM25	0.310	0.315	+1.61%	$b = 0.2$
F2EXP	0.289	0.297	+2.77%	$s = 0.2^*$
F2LOG	0.295	0.301	+2.03%	$s = 0.3^*$
BM3	0.316	0.295	-6.65%	$\mu = 2700$
BM25+	0.318	0.318	+0.00%	$b = 0.2$ $\delta = 1.0$
PIV and its variants				
PIV	0.292	0.295	+1.03%	$s = 0.1$
F1EXP	0.288	0.278	-3.47%	$s = 0.0^*$
F1LOG	0.288	0.277	-3.82%	$s = 0.0^*$
PIV+	0.295	0.299	+1.36%	$s = 0.01$ $\delta = 0.4$
Language modeling approaches				
DIR	0.294	0.310	+5.44%	$\mu = 3000$
TSL	0.278	0.312	+12.23%	$\mu = 3500^*$ $\lambda = 0.0^*$
F3EXP	0.288	0.290	+0.69%	$s = 0.05^*$
F3LOG	0.290	0.293	+1.03%	$s = 0.05^*$
DIR+	0.312	0.312	+0.00%	$\mu = 3000^*$ $\delta = 0.01$
Divergence from Randomness Models				
PL3	0.293	0.288	-1.71%	$\mu = 9700$
PL2+	0.326	0.327	+0.31%	$c = 23$ $\delta = 0.8$

Table 4.7: Performance comparison of reproduced and original results on **disk4&5**

RM	**orig.**	**repd.**	**diff.**	**para.**
BM25 and its variants				
BM25	0.254	0.247	-2.76%	$b = 0.4$
BM3	0.251	0.238	-5.18%	$\mu = 950$
BM25+	0.255	0.249	-2.35%	$b = 0.4$ $\delta = 1.0$
PIV and its variants				
PIV	0.241	0.221	-8.30%	$s = 0.05$
PIV+	0.246	0.238	-3.25%	$s = 0.5$ $\delta = 0.01$
Language modeling approaches				
DIR+	0.253	0.252	-0.40%	$\mu = 1000^*$ $\delta = 0.01$
Divergence from Randomness Models				
PL3	0.230	0.239	+3.91%	$\mu = 1600$
PL2+	0.254	0.255	+0.39%	$c = 9$ $\delta = 0.8$
Information-based Models				
LGD	0.250	0.251	+0.40%	$c = 2.0$
SPL	0.254	0.251	-1.18%	$c = 9.0$

Table 4.8: The mean and standard deviation of the performance difference between the reproduced and original results

Functions	**Mean**	**Std.**
BM25 and its variants		
BM25	-2.08%	4.11%
F2EXP	+0.68%	2.18%
F2LOG	+0.22%	1.63%
BM3	-5.92%	0.74%
BM25+	-0.67%	1.19%
PIV and its variants		
PIV	-3.64%	4.67%
F1EXP	-6.62%	2.23%
F1LOG	-7.76%	2.79%
PIV+	-0.94%	2.31%
NTFIDF	-17.08%	4.71%
Language modeling approaches		
DIR	+1.03%	3.26%
TSL	+4.09%	6.18%
F3EXP	-2.65%	2.72%
F3LOG	-4.11%	3.74%
DIR+	-0.20%	0.20%
Divergence from Randomness Models		
PL2	+5.54%	17.73%
PL3	+0.59%	2.41%
PL2+	+0.35%	0.04%
Information-based Models		
SPL	-4.60%	3.42%
LGD	-2.04%	2.45%

Table 4.9: Reproduced performance comparison for PL2 and NTFIDF

Functions	**collections**	**orig.**	**repd.**	**diff.**	**para.**
PL2	TREC1	0.207	0.257	+24.46%	$c = 1.0$
	TREC2	0.238	0.285	+19.60%	$c = 1.0$
	TREC3	0.271	0.327	+20.89%	$c = 1.0$
	TREC6	0.257	0.233	-9.30%	$c = 1.0$
	TREC7	0.221	0.196	-11.39%	$c = 1.0$
	TREC8	0.256	0.228	-11.01%	$c = 1.0$
NTFIDF	TREC678	0.234	0.209	-10.64%	
	ROBUST04	0.302	0.245	-18.84%	
	GOV2	0.317	0.248	-21.77%	

Table 4.10: Optimal MAP/ERR@20 for all collections. * indicates the model is significant better than the base model in its category (always the first one). † indicates the model is the best performed in its category. ‡ indicates the model is significant better than all other models in its category. All significant tests are at p = 0.05 by a paired one-tailed t-test.

RM	**disk12**	**disk45**	**WT2G**	**GOV2**	**CW09**	**CW12**
BM25 and its variants						
BM25	0.204	0.248	0.315	0.297	0.089	0.128
F2EXP	0.228*	0.251	0.297	0.284	0.099*	0.139†
F2LOG	0.231*	0.252†	0.302	0.297	0.100*	0.137
BM3	0.234*	0.241	0.296	0.283	0.098*	0.130
BM25+	0.235*†	0.249	0.318†	0.301†	0.102*†	0.137
PIV and its variants						
PIV	0.201	0.221	0.294	0.254	0.104	0.137
F1EXP	0.198	0.221	0.278	0.240	0.100	0.135
F1LOG	0.200	0.217	0.277	0.255	0.104	0.137
PIV+	0.207*†	0.239*‡	0.299*	0.265*	0.113†	0.141†
NTFIDF	0.205*	0.213	0.307*†	0.296*‡	0.097	0.129
Language Modeling Approaches						
DIR	0.227	0.252†	0.312	0.299	0.090	0.134
BLM	0.208	0.233	0.314†	0.222	0.072	0.113
TSL	0.228†	0.252	0.312	0.300†	0.090	0.134
F3EXP	0.205	0.234	0.290	0.250	0.101*	0.138
F3LOG	0.203	0.232	0.293	0.263	0.109*†	0.138†
DIR+	0.227	0.252	0.312	0.299	0.090	0.134
Divergence from Randomness Models						
PL2	0.228	0.252	0.325	0.303†	0.089	0.116
PL3	0.228†	0.241	0.290	0.269	0.093†	0.117
PL2+	0.214	0.255*‡	0.328*‡	0.301	0.089*	0.119*†
Information-based Models						
LGD	0.215†	0.251†	0.320†	0.300†	0.086	0.131†
SPL	0.213	0.251	0.313	0.299	0.093†	0.130

Table 4.11: Free Parameters used in Parameter Tuning

Model	**Para. Range**	**Incr.**
BM25	$b \in [0, 1]$	0.05
PIV, F1EXP, F1LOG, F2EXP, F2LOG, F3EXP, F3LOG	$s \in [0, 1]$	0.05
DIR, BLM,	$\mu \in [500, 5000]$	500
TSL	$\mu \in [500, 5000]$ $\lambda \in [0, 1]$	500 0.1
PL2	$c \in [0.5] \cup [1, 25]$	1
BM3, PL3	$c \in [0.5] \cup [0.75] \cup [1, 9]$ $\mu \in [500, 5000]$	1 500
BM25+	$b \in [0, 1]$	0.05
PIV+	$s \in [0, 1]$	0.05
DIR+	$\mu \in [500, 5000]$	500
PL2+	$c \in [0.5] \cup [1, 25]$	1
BM25+, PIV+, DIR+, PL2+	$\delta \in [0.0, 1.5]$	0.1

Chapter 5

TOOLS FOR UNDERSTANDING THE EXISTING IR RANKING MODELS AND KEYWORD QUERIES

Various information retrieval models have been studied for decades. Most traditional retrieval models are based on bag-of-term representations, and they model the relevance based on various collection statistics, e.g. Term Frequency (TF), Inverted Document Frequency (IDF), Document Length (DL). Bag-of-term document representation and the statistics could be treated as the context of such ranking models. Despite these efforts, it seems that the performance of "bag-of-term" based retrieval functions has reached plateau, and it becomes increasingly difficult to further improve the retrieval performance. Thus, one important research question is whether we can provide any theoretical justifications on the empirical performance bound of basic retrieval functions.

In our work, we start with single term queries, and aim to estimate the performance bound of retrieval functions that leverage only basic ranking signals. Specifically, we demonstrate that, when only single term queries are considered, there is a general function that can cover many basic retrieval functions. We then propose to estimate the upper bound performance of this function by applying a cost/gain analysis to search for the optimal value of the function.

It is well known that query formulation could affect retrieval performance. Empirical observations suggested that a query may contain extraneous terms that could harm the retrieval effectiveness. This is true for both verbose and title queries. Given a query, it is possible that using its subqueries can generate more satisfying search results than using the original query. Although previous studies proposed method to reduce verbose queries, it remains unclear how we could reduce title queries given the short

Table 5.1: Instantiations of the general retrieval form

Retrieval Functions	$g(\cdot)$	α	c_1	γ	β	c_2
DIR	1	1	$\mu \cdot p(t\|C)$	0	1	μ
BM25 & BM25+	1	$k_1 + 1$	0	1	$\frac{k_1 \cdot b}{avdl}$	$k_1 \cdot (1-b)$
PIV & PIV+	$1 + ln(1 + ln(\cdot))$	1	0	0	$\frac{s}{avdl}$	$1 - s$
F1EXP & F1LOG	$1 + ln(1 + ln(\cdot))$	$avdl + s$	0	0	s	$avdl$
F2EXP & F2LOG	1	1	0	1	$\frac{s}{avdl}$	s
BM3	1	1	$\mu \cdot p(t\|C)$	μ	k_1	$k_1 \cdot \mu + \mu^2 \cdot p(t\|C)$
DIR+	1	$\mu \cdot p(t\|C) + \delta$	$\mu^2 \cdot p^2(t\|C) + \delta \cdot \mu \cdot p(t\|C)$	0	$\mu \cdot p(t\|C)$	$\mu^2 \cdot p(t\|C)$

length of the title queries. In this thesis, we focus on identifying the best performed subqueries for a given query. In particular, we formulate this problem as a ranking problem, where the goal is to rank subqueries of the query based on its predicted retrieval performance. To tackle this problem, we propose a set of novel post-retrieval features that can better capture relationships among query terms, and apply a learning-to-rank algorithm based on these features. Empirical results over TREC collections show that these new features are indeed useful in identifying the best subqueries.

5.1 Performance Bound Analysis for Single Term Queries

5.1.1 Introduction

Developing effective retrieval models has been one of the most important and well-studied topics in Information Retrieval (IR). Various retrieval models have been proposed and studied [87, 92, 118]. Many of them are based on "bag-of-term" representation and leverage only basic ranking signals such as TF, IDF and document length normalization [29]. Although more advanced ranking signals, such as term proximity [94] and term semantic similarity [29, 66], have been integrated into the retrieval functions to improve the retrieval performance, it remains unclear whether we have reached the performance upper bound for retrieval functions using only basic ranking signals. If so, what is the upper bound performance? If not, how can we do better?

To find the performance upper bound is quite challenging: although most of the IR ranking models deal with basic signals, how they combine the signals to compute the relevance scores are quite diverse due to different implementations of IR heuristics [29]. This kind of variants makes it difficult to generalize the analysis. Moreover, typically

there are one or more free parameters in the ranking models which can be tuned via the training collections. These free parameters make the analysis more complicated.

This thesis aims to tackle the challenge through the simplest problem setup. In particular, we focus on single-term queries and study how to estimate the performance bound for retrieval functions utilizing only basic ranking signals. With only one term in a query, many retrieval functions can be greatly simplified. For example, Okapi BM25 and Pivoted normalization functions have different implementations for the IDF part, but this part can be omitted in the functions for single-term queries because it would not affect the ranking of search results. All the simplified functions can then be generalized to a general function form for single-term queries. As a result, the problem of finding the upper bound of retrieval function utilizing basic ranking signals becomes that of finding the optimal performance of the generalized retrieval function. We propose to use cost/gain analysis to solve the problem [10,11,28]. As the estimated performance upper bound of simplified/generalized model is in general better than the existing ranking models, our finding provides the practical foundation of the potentially more effective ranking models for single term queries.

5.1.2 A General Form of Retrieval Functions for Single-Term Queries

The implementations of retrieval functions are quite diverse, and it is often difficult to develop a general function form that can cover many retrieval functions. However, if we consider only single-term queries (i.e,. those with only one query term), the problem can be greatly simplified.

Let us start with a specific example. Dirichlet prior function is one of the representative functions derived using language modeling approaches [118], and is shown as follows:

$$f(Q,d) = \sum_{t \in Q} \ln\left(\frac{c(t,d) + \mu \cdot p(t|C)}{|d| + \mu}\right), \tag{5.1}$$

where $c(t,d)$ is the frequency of term t in document d, $|d|$ is the document length; $p(t|C)$ is the maximum-likelihood of the term frequency in the collection and μ is the

model parameter. When a query contains only a term t, the retrieval function can be simplified to:

$$f(\{t\}, d) = \frac{c(t,d) + \mu \cdot p(t|C)}{|d| + \mu} \tag{5.2}$$

Note the natural logarithm function in Equation (5.1) is omitted since it is a monotonically increasing function and would not affect the ranking results. Since $p(t|C)$ is a collection-dependent constant, the function can be further simplified as:

$$f(t,d) = \frac{c(t,d) + c_1}{|d| + c_2}. \tag{5.3}$$

Similarly, Okapi BM25 [87] can be simplified to:

$$\begin{aligned} f(t,d) &= \frac{(k_1+1) \cdot c(t,d)}{c(t,d) + k_1 \cdot (1 - b + b \cdot |d|/avdl)} \\ &= \frac{\alpha \cdot c(t,d)}{c(t,d) + \beta \cdot |d| + c_2}, \end{aligned} \tag{5.4}$$

where α absorbs $k_1 + 1$, and $\beta = k_1 \cdot b/avdl$ is a collection-dependent variable and $c_2 = k_1 \cdot (1 - b)$ is a parameter.

Furthermore, the pivoted normalization function (PIV) [92] can also be simplified to:

$$\begin{aligned} f(t,d) &= \frac{1 + ln(1 + ln(c(t,d)))}{(1 - s + s \cdot |d|/avdl)} \\ &= \frac{g(c(t,d))}{(\beta \cdot |d| + c_2)}, \end{aligned} \tag{5.5}$$

where $g(\cdot) = 1 + ln(1 + ln(\cdot))$ and can be further generalized as an arbitrary non-linear function. $\beta = s/avdl$ is a collection related variable and $c_2 = 1 - s$ is a parameter.

All of the above three simplified functions (i.e., Eq. (5.3), Eq. (5.4) and Eq. (5.5)) can be generalized as the following form:

$$F(c(t,d), |d|) = \frac{\alpha \cdot g(c(t,d)) + c_1}{\gamma \cdot c(t,d) + \beta \cdot |d| + c_2}, \tag{5.6}$$

where $g(\cdot)$ is an arbitrary non-linear function and $\alpha, \beta, \gamma, c_1, c_2$ are free parameters. This generalized function form is essentially a linear transformation of a non-linear transformation of term frequency divided by a linear transformation of document length.

The denominator optionally adds adjusted term frequency as a method to dampen the impact of increasing term frequency. Note that IDF is not part of the function because it would not affect the document ranking for single-term queries.

In fact, we find that the generalized retrieval function as shown in Eq. (5.6) can cover at least 11 retrieval functions. In addition to the above three retrieval functions, the following functions can also be generalized: (1) F1EXP, F1LOG, F2EXP and F2LOG from the axiomatic retrieval models [33], (2) BM3 derived from the Dirichlet Priors for term frequency normalization model [43], and (3) BM25+, DIR+, PIV+ derived from the lower bounding term frequency normalization models [69]. Table 5.1 summarizes the instantiations for each of the retrieval functions.

5.1.3 Upper Bound Estimation for MAP

Given the general form as shown in Equation (5.6), one straightforward solution to estimate the performance bound for single-term queries would be to simply try all possible values/instantiations for the parameters and functions and then report the best performance. Thus, the problem of estimating performance bound boils down to the problem of searching for optimal parameter settings in terms of the retrieval performance. More specifically, given Eq. (5.6), we need to find parameter settings for $\alpha, \beta, \gamma, c_1, c_2$ that can optimize the retrieval performance measured (i.e., MAP in this thesis). We do not consider the instantiation of $g(\cdot)$ here, and leave it as our future work.

Since it is infeasible to try all possible parameter values and find the optimal setting, we propose to apply the cost/gain analysis to find the optimal parameter setting.

Let us explain the notations first. d_i and d_j are a pair of documents. Given a query, $s_i = f(\{t\}, d_i)$ and $s_j = f(\{t\}, d_j)$ denote the relevance score of these two documents computed using a retrieval function.

For a given query, each pair of documents d_i and d_j with different relevance labels (currently we only consider the binary case, i.e. whether the document is relevant or

non-relevant) a ranking model computes the scores $s_i = f(d_i)$ and $s_j = f(d_j)$. Follow the previous studies about RankNet [10,11], we define the cost function as the pairwise cross-entropy cost applied to the logistic of the difference of the relevance scores:

$$C_{ij} = \frac{1}{2}(1 - S_{ij})\sigma(s_i - s_j) + \log(1 + e^{-\sigma(s_i - s_j)}) \tag{5.7}$$

where $S_{ij} \in \{0, \pm 1\}$ denotes the ground-truth ranking relationship of document pair d_i and d_j: 1 if d_i is relevant and d_j is non-relevant, -1 if d_i is non-relevant and d_j is relevant, 0 if they have the same label. The gradient of the cost function is then:

$$\frac{\partial C_{ij}}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right) = -\frac{\partial C_{ij}}{\partial s_j} \tag{5.8}$$

If we only consider the total cost of ranking non-relevant documents before the relevant documents, S_{ij} is always 1. We will always consider that d_i is relevant and d_j is non-relevant from now on. The Eq. (5.8) is then simplified as:

$$\frac{\partial C_{ij}}{\partial s_i} = \frac{-\sigma}{1 + e^{\sigma(s_i - s_j)}} \tag{5.9}$$

The upper bound of the performance is then obtained when the cost is minimized by parameters optimization. The parameters $\mathrm{p_k} \in \mathbb{R}$ used in the ranking model could be updated so as to reduce the cost via stochastic gradient descent:

$$\mathrm{p_k} \rightarrow \mathrm{p_k} - \eta\frac{\partial C}{\partial \mathrm{p_k}} = \mathrm{p_k} - \eta\left(\frac{\partial C}{\partial s_i}\frac{\partial s_i}{\partial \mathrm{p_k}} + \frac{\partial C}{\partial s_j}\frac{\partial s_j}{\partial \mathrm{p_k}}\right) \tag{5.10}$$

Unfortunately, the cost defined in Eq. (5.9) is actually the "optimization" cost instead of the target cost (the actual cost) [11] and thus minimizing the cost may not necessarily lead to the optimal MAP. However, MAP is either flat or non-differentiable everywhere which makes the direct optimization toward it difficult [116]. To overcome this we modify Eq. (5.9) by multiplying the derivative of the cost by the size of the change in MAP gain from swapping a pair of differently labeled documents for a given query q. The pairwise λ (we change cost C to λ and λ is the gain instead of cost) can be written as:

$$\lambda_{ij} = \frac{\sigma}{1 + e^{\sigma(s_i - s_j)}}\frac{1}{|R|}\left(\left|\frac{n}{r_j} - \frac{m}{r_i}\right| + \sum_{k=r_j+1}^{r_i-1}\frac{\mathrm{I}(k)}{k}\right) \tag{5.11}$$

Table 5.2: collections and queries

	disk12	Robust04	WT2G	GOV2
#queries	4	11	3	2
qid	57,75,77,78	312,348,349,364,367,379, 392,395,403,417,424	403,417,424	757,840

where r_i and r_j are the ranking positions of d_i and d_j; m and n are the number of relevant documents before position r_i and r_j; $\mathrm{I}(k) = 1$ if the document at kth position of the ranking list is relevant and 0 otherwise; $|R|$ is the number of relevant document for the query. The model parameters are adjusted based on the aggregated λ for all pairs of documents for the query using a small (stochastic gradient) step.

The optima are local optima with 99% of the confidence by following the Monte-Carlo method with model parameters chosen from 459 random directions [28].

5.1.4 Experiments

5.1.4.1 Testing Collections

We use four TREC collections: disk12, Robust04, WT2G and Terabyte (GOV2) to conduct the experiments. For the queries, only the title fields of the query topics with only one query term are used (20 in total). We use Dirichlet language model with default $\mu = 2500$ to retrieve at most top 10,000 documents as the documents pool for the pairwise comparison for each query. For relevance labels that less or equal to zero is treated as non-relevant and labels greater than zero are treated as relevant. An overview of the involved collections and queries are listed in Table 5.2.

5.1.4.2 Experiment Setup

We tested both using the cost function only and using the cost function together with λ component of MAP. The results are very close and the cost with λ seems to be a little bit superior so we just report that part of the results. We basically tried several different models based on Eq. (5.6):

- **DIR**^{U}: Dirichlet Language Model, denoted as $\frac{c(t,d)+\mu \cdot p(t|C)}{|d|+\mu}$
- **TFDL1**^{U}: which only contains c_1 and c_2 as model parameters, denoted as $\frac{c(t,d)+c_1}{|d|+c_2}$

Table 5.3: Upper Bound of MAP

		disk12	Robust04	WT2G	GOV2
Models with Basic Signals	DIR	0.4009	0.3823	0.3660	0.2083
	BM25	**0.4016**	**0.3824**	**0.4038**	0.2896
	PIV	0.3987	0.3812	**0.4038**	**0.3079**
	F2EXP	0.4000	0.3682	0.3183	0.1950
	BM3	0.4015	0.3823	0.3792	0.2554
	DIR+	0.4009	0.3823	0.3794	0.2083
Upper Bounds	DIR^U	$0.4244^\dagger$	$0.4136^\dagger$	0.4055	0.2724
	$TFDL1^U$	$0.4273^\dagger$	$0.4209^\dagger$	0.4095	$0.3193^\dagger$
	$TFDL2^U$	$\mathbf{0.4273}^\dagger$	$\mathbf{0.4209}^\dagger$	**0.4095**	$\mathbf{0.3255}^\dagger$

- **TFDL2**^{U}: which takes α, β, c_1, c_2 as parameters, denoted as $\frac{\alpha \cdot c(t,d)+c_1}{\beta \cdot |d|+c_2}$

For other possible format of Eq. (5.6) they are essentially covered by $TFDL2^U$ so we do not report the results for them[1].

For all of our experiments, we varied the learning rate η between 10^0 to 10^{10} with step size 10 times to previous value. We have found that optimal learning rate brings marginal gain in terms of overall performance. So we just report the performance on the optimal learning rate. For the starting point, we choose α, β, c_1, c_2 from $[0.1, 10000]$ with step size 10 times to previous value. We set the learning iteration at most 500 epochs and it stops if the gain was constant over 20 epochs.

5.1.4.3 Results

Table 5.3 lists both the optimal performances of previously proposed ranking models with optimal parameters chosen from a wide range (e.g. for DIR and DIR+ $\mu \in [0, 5000]$ with step size 500; for BM25, BM3, PIV, F2EXP b or $s \in [0, 1]$ with step size 0.1) and optima of proposed models. The values listed in the table are the MAPs of single term queries only (not the whole set of the queries). It is shown that the generalized models are better than classic ranking models for the most cases (indicated by the † which means the two-tailed paired t-test at p value of 0.05 comparing

[1] Actually they are possibly covered by Eq. (5.6). But if we choose wide spectrum of the starting points then they are covered by large chance.

Table 5.4: Parameters

Model	Paras	disk12	Robust04	WT2G	GOV2
DIR^U	μ	4.66e3	3.54e7	1.43e6	0
TFDL1^U	$\frac{c_1}{c_2}$	2.49e-3	1.0	6.87e-5	6.0e-1
TFDL2^U	$\frac{c_1}{c_2}$	1.55e-2	5.86e-2	1.08e1	1.39e-1
	$\frac{\alpha}{\beta}$	1.37e-4	1.43e-2	1.01e-2	1.13e-2

with the optimal performances of selected models which are boldfaced). Furthermore, different collections have different gains. Robust04 has the largest gain between the two results which indicating that possibly the previously proposed ranking models do not capture the critical ranking signals well or the statistics they use contradicts with the actual properties of relevant documents. Also, for WT2G we get very little gain by applying our analysis (the performances are even not significant better than the selected models). This probably means that if we would like to further improve the performance on WT2G we need to find other forms of the ranking models which may look different than Eq. (5.6).

5.1.4.4 Parameters

Next, we would like to investigate the parameters that lead to the optima for the proposed models. The parameters are worthy to look at since they might inspire or provide intuition of better performing models in the future. Table 5.4 lists those parameters. As we can see, for DIR^U the optimal parameters μ obtained for Robust04 and WT2G are much larger than 10^3 which is suggested value by the original authors of DIR [118]. For TFDL1^U we choose to report the ratio $\frac{c_1}{c_2}$. The values vary between collections. For example, the optimal values for Robust04 is 1.0 which indicates that the better performed models would have larger dampen factor for document length than other collections. For TFDL2^U both $\frac{c_1}{c_2}$ and $\frac{\alpha}{\beta}$ are reported. We find that α is in several magnitude levels smaller than β. But this is not always the truth for $\frac{c_1}{c_2}$. We would expect more impact on $\frac{\alpha}{\beta}$ than $\frac{c_1}{c_2}$ and the values of $\frac{\alpha}{\beta}$ could be better incorporated by better performing models in the future.

5.2 Reducing the Keyword Queries

5.2.1 Introduction

The retrieval performance is closely related to the quality of a query. Not all query terms in a query are equally important, and sometimes removing a term from the query might lead to better retrieval performance.

The problem of query reduction has been studied intensively for verbose queries (i.e., those are formulated based on the description of TREC topics) [5,31,52]. Previous studies showed that although a subquery does not always perform as well as the original query, the best subquery could be much better – 23% improvement in terms of MAP for verbose queries [5,52]. However, reducing keyword queries (e.g. those formulated based on the title of TREC topics) has drawn less attention than its counterpart. Previous study [31] showed that *title queries can also be reduced to obtain better performances* on ClueWeb collection. We made similar observations based on the results on other TREC collections too. Table 5.5 compares the performance when using the original keyword queries with those using the best performed subqueries. Although the table only contains the queries with length 3, it can be seen that the performance of using the best subqueries has more than 10% of gain in terms of the effectiveness. It is clear that *reducing keyword queries could lead to better performance*, but the problem is how to identify the best-performed subquery for a given query when we do not have any information about the relevance judgments.

Reducing keyword queries is a challenging task. Given a keyword query is already very short, how can we ever remove terms from that? One simplest solution would be to remove the terms based on their IDF values. Unfortunately, it does not work well. Let us consider query "pheromone scents work" (from WT10G). Among all the query terms, "work" has the lowest IDF. However, removing "work" from the query would not achieve our goal since the best-performed subquery for this query is "pheromone work". Similarly, other features, such as mutual information and clarity score, which were successfully applied in reducing verbose queries, are not as useful as what is shown in the previous work [52].

In this thesis, we focus on the problem of identifying best-performed subquery for a given keyword query. In particular, we formulate the problem as ranking all the subqueries of a keyword query based on their predicted performance. To tackle this problem, we propose a set of novel features that can better capture the relations among query terms, and then apply a learning-to-rank algorithm to rank the subqueries based on these new features as well as some existing ones.

All the proposed new features are post-retrieval ones, meaning that they are computed based on the retrieval results. These features are designed to capture different relationships among query terms from different aspects: query term proximity, the aggregated ranking scores of query terms, and the compactness and position of term tensors. Specifically, *term proximity based features* are designed to capture the intuition that some query terms should be viewed as phrases as opposed to individual terms. Let us consider query "family leave law". Its best subquery is "family leave", which is a law code. And only when the two terms occur next to each other and in the right order in a document, we are sure that the document is relevant. To capture this intuition, we propose to compute the statistics of the ranking scores that are computed based on term dependency model [72] for each subquery, and also based on the correlations between these ranking scores with the ranking scores of the original query. Furthermore, we proposed another set of *term score based* features that are designed to measure the balance between TF and IDF weighting [30]. These features are computed based on different ways of aggregating the term scores of individual query terms. The assumption is that these statistics could capture the key properties of the best subquery at the term score level and thus are useful. Finally, we proposed a set of features based on the *compactness and positions of the term score tensors*. For this set of features , we investigate the spatial properties of the term scores. We view the term scores from top ranked documents as tensors in the multi-dimensional space and then compute the compactness and the position of the tensors cluster.

Empirical results show that the proposed new features are effective in identifying the best-performed subquery. Moreover, we intensively analyze the important features

Table 5.5: Comparison of the MAP between using original queries and optimal subqueries. Only queries of length 3 are shown and the ranking function is BM25

Collection	Original	Upper Bound	Diff.
Disk12	0.2597	0.2880	+10.9%
Disk45	0.2399	0.2772	+15.5%
AQUAINT	0.2107	0.2426	+15.1%
WT2G	0.3285	0.3580	+9.0%
WT10G	0.1720	0.2051	+19.2%
GOV2	0.3060	0.3221	+5.3%

by comparing the performance difference between the subset of features and all features. The results validate the utility of the proposed new features.

5.2.2 Subquery Ranking Details

In this section we present details of how we identify the best-performed subquery for a give keyword query by ranking all possible subqueries.

5.2.2.1 Problem Setup

The best-performed subquery identification problem can be formally defined as follows. Given an arbitrary query $Q = \{t_1, t_2, ..., t_{|QL|}\}$, let P^Q denote the power set of Q. Let f be a retrieval function used for ranking the documents in the collection for any query in P. Let $m(P, f)$ denote a metric for the ranking effectiveness of retrieval function f using query P. The best-performed subquery identification problem aims at finding a subquery $P^* = \arg\max_{P \in P^Q} m(P, f)$.

5.2.2.2 Subquery Ranking

We formulate the best-performed subquery identification problem as a subqueries ranking problem. The ranking is based on the predicted performance of the subqueries without prior knowledge of relevance. We feed the features of the subqueries to the state-of-the-art learning-to-rank algorithm LambdaMART [10] where the ranking of the subqueries is obtained and the best subquery is identified.

5.2.2.3 Subquery Ranking Features

Term Relationship Features

We introduce the newly proposed features that can better capture the relations among query terms – the motivation behind them as well as the detailed steps to compute them. These features are designed to capture different relationships among query terms from different aspects: query term proximity, the aggregated ranking scores of query terms, and the compactness and position of term tensors. The above mentioned features are *post-retrieval features* where we explore the scores in the ranking list of the subquery and generate the features from that. The variables and the notations that will be used in the following sections are summarized in Table 5.6.

Term Proximity Based Features (PXM)

Term proximity based features are mainly inspired by the term dependency model [72]. The intuition is that some query terms should be treated as phrases instead of separated terms. Consider the original query $Q = \{\text{family leave law}\}$ (ROBUST04,qid:648), the best subquery of it is $P^* = \{\text{family leave}\}$. In this example, "family leave" is supposed to refer the law code and only when the two terms occur together and in the exact order in a document, the document is possibly relevant. We propose to leverage the ranking scores which are computed using the term proximity model from top ranked documents and extract the high level statistics from the scores. Presumably, the term proximity model introduce more restrict requirement of the matching documents than the original query and thus the ranking list might contain fewer documents. By only looking at the top ranked documents we increase the probability of having the same number of documents in the ranking list for all subqueries. Also, based our experience the ranking scores of term proximity model usually have larger standard deviation.

For detailed steps, we first rank the documents in the collection using our customized term proximity models (see example query below). We have tried three models - unordered window model (UW), ordered window model (OW) and the combination

Table 5.6: Notations and Explanations

Notations	Explanations
$Q = \{t_1, t_2, ..., t_{\|QL\|}\}$	The original query and its terms
$\|QL\|$	Query length
$P^Q = \{q_1, q_2, ..., q_i, ...\}$	The power set of Q which contains all subqueries
q	The general notation for any subquery including the original query.
c	Ranking list cutoff position
d_i	Document i in the ranking list
$ds_{q,i}$	Ranking score of document i for query q
$L_{q,c}(f) = \{d_1, ..., d_c\}$	Ranking list of q using model f cutoff at c.
$\vec{SL_{q,c}}(f) = \{ds_{q,1}, ..., ds_{q,c}\}$	Ranking scores in $L_{q,c}(f)$.
$\vec{t_{q,d_i}}(f) = \{f_{t_1,d_i}, ..., f_{t_n,d_i}\}$	Terms scores of d_i for query q computed by model f. $n = \|QL\|$.
$\vec{TL_{t_i,c}}(f) = \{f_{t_i,d_1}, ..., f_{t_i,d_c}\}^T$	Column term scores for t_i.
$ML_{q,c}(f) = \{\vec{t_{q,d_1}}(f), ..., \vec{t_{q,d_c}}(f)\}$	Terms scores matrix of $L_{q,c}(f)$.
$g(\vec{x}), h(\vec{x}) \in \Re$	Feature function. One of **MIN**, **MAX**, **MAX-MIN** (difference), **MAX/MIN** (division), **SUM**, **MEAN**, **STD** (standard deviation), **GMEAN** (geometric mean)

of the two models (UWOW). The window parameter (i.e. terms must appear with at most how many terms between each) wd is set to $4 \cdot (|QL| - 1)$. The sample query of UWOW using Indri query language would be

```
#combine(#uw4(family leave) #ow4(family leave))
```

Then we extract the high level statistics from the document scores at cutoff c as the features by applying the feature functions $h(\vec{x})$ to the scores. The feature function h is defined in Table 5.6 and it consists of a set of statistical functions that can be applied to a vector of values such as summation and standard deviation. The use of feature function was shown to be beneficial in order of aggregating the raw values in previous

works [8, 23]. Formally, we have

$$PXM(w)_h = h(\vec{SL_{q,c}}(w)) \tag{5.12}$$

where $w \in UW, OW, UWOW$ and $\vec{SL_{q,c}}(w)$ is the documents scores vector of the term proximity model. Only focusing on the properties of the ranking scores of term proximity model may not be enough – it is probably beneficial to also consider the the normal (no term proximity involved) ranking scores of the original query as well. By comparing the the scores in the two ranking lists we might have more insights about the subqueries. Presumably, if a specific subquery performs much better than the original query because of the subquery is the key phrase in the original query then the scores of the two ranking lists should be different from each other. Take our previous subquery "family leave" for example, our method assumes that this subquery (actually the term proximity model) should have different ranking scores from the original query "family leave law" ranking scores and we are expected to capture such feature. Based on the above reasoning we measure the correlation between the documents scores of the term proximity model of the subquery and the regular ranking scores of the original query Q using Kendall's Tau (τ_B) and Pearson's r. Formally,

$$PXM(w)_{corr} = Corr(\vec{SL_{Q,c}}(w), \vec{SL_{q_i,c}}(w)) \tag{5.13}$$

where $Corr \in \{\tau_B, \rho\}$.

Term Score Based Features (TS)

For this set of features, we continue to explore the ranking list of the subqueries – the scores of individual query terms instead of the score of the document. The intuition of TS is originated from the term frequency constraint and the term discrimination constraint from previous work [30]. The constraint essentially introduces the balance between document term frequency (TF) and inverted document frequency (IDF). This really inspires us that there should be some interesting properties in the term score of top ranked documents. Instead of separately considering the TF and IDF we choose to directly look at the individual term score computed by any ranking function that has

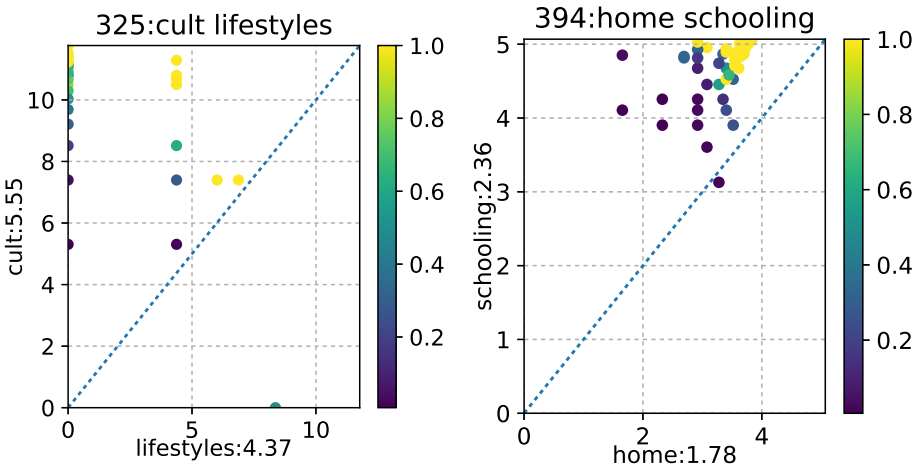
reasonable TF and IDF components (please refer to [30] for a more comprehensive list of the reasonable ranking functions) for two reasons: (1) the ranking list is determined by the scores computed by the ranking function, and (2) the ranking function has the TF and IDF components and thus it already naturally adopts the constraint mentioned in [30]. We wonder, for instance, do the top ranked documents have more balanced term scores or do they have highly skewed term scores? Or is the performance of subquery related to the minimum of the term scores in the top ranked documents? Figure 5.1 illustrates the intuition: the two subfigures are the term scores computed by BM25 model for the two queries for TREC keyword topics. From the figures we find two distinct patterns: for query "cult lifestyle" its relevant documents have higher probability along the y-axis indicating that "cult" is more important than "lifestyle" for this query. But for query "home schooling" the term scores are more balanced. For TS features we calculate the score for each subquery term in the top ranked documents using any ranking function that has reasonable TF and IDF components (for both document and term), e.g. BM25 or language model with Dirichlet. We then generate high-level statistics from the term scores.

The TS features are then computed as follows: we first generate the ranking list $L_{q,c}\vec{(f)}$ for subquery q using any ranking function that has reasonable TF and TD components. For each document d_i in $L_{q,c}\vec{(f)}$ we compute the score for each individual term. This would generate the term scores vector $t_{q,d_i}\vec{(f)}$ for d_i. We then apply the feature function h to $t_{q,d_i}\vec{(f)}$ to get the aggregated statistics for d_i as $h(t_{q,d_i}\vec{(f)})$. The result of this step is a list of statistics with each element corresponding to one document. We then apply the feature function g again to each column of the previously generated list $h(t_{q,d_i}\vec{(f)})$ to generate the final TS features. Formally, TS is computed as:

$$TS(f,h,g) = g(h(t_{q,d_i}\vec{(f)})) \tag{5.14}$$

For example, $TS(BM25, MEAN, STD)$ for query "home schooling" first rank the documents in the collection using BM25 function and then we compute the terms scores for "home" and "schooling" for each document in the ranking list again using

Figure 5.1: Individual term scores. Term scores are computed using BM25 model. Colors of the dots are the probability of relevant document at that point. Axis labels show the IDF values computed by $log\frac{N}{df}$.



(a) AQUAINT QID:325

(b) AQUAINT QID:394

BM25 function. The average value of terms scores for each document in the ranking list is then calculated and this results in a list of average values. Finally the standard deviation of the average values is computed and the value is served as the feature.

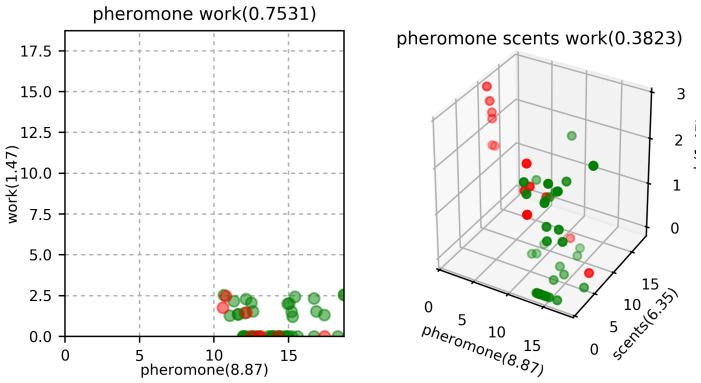
Compactness and Positions of Term Score Tensors (TCP)

Document score in the ranking list as the query prediction feature was previously proposed in [90]. In their work, the feature Normalized Query Commitment (NQC) which is essentially defined as the standard deviation of document scores in the ranking list was used as a post-retrieval feature to predict the query performance. Higher deviation value was correlated with potentially lower query drift, and thus indicating the better effectiveness [90]. We also find the deviation and other statistics of ranking scores are indeed useful. However, our proposed features are different from that of [90] in the sense that we focus on the term level scores instead of document level scores.

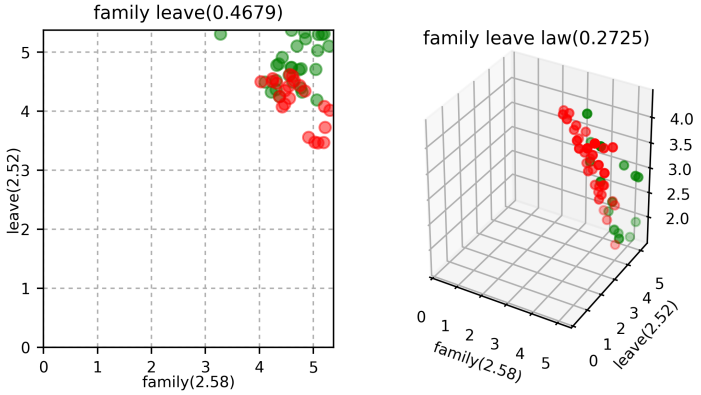
Figure 5.2 shows two example queries from WT10G and ROBUST04 respectively. The x-axis and y-axis are the term scores computed by BM25 model[2] and only

[2] Using Dirichlet language model would get the similar conclusion albeit the values

Figure 5.2: Terms scores (computed by BM25) of the top 50 ranked documents in the list. The numbers in the titles are the Average Precision of the corresponding subquery. Green dots are relevant documents and red dots are non-relevant documents. For each query only the optimal subquery and the original query are shown.



(a) WT10G QID:530



(b) ROBUST04 QID:648

the top 50 ranked documents are included in the figures. For both queries, the best subqueries are the queries with fewer query terms, i.e. not the original query. We find the similarity and difference for the chosen queries. First, it can be seen that for both best subqueries the term scores from top ranked documents are more compactly clustered. Second, the two queries are different in the sense that the term scores clusters are located at the different position in the two dimensional space. Such difference indicates that for the best subquery the ranking model has its own preference

---

are negative.

among query terms. For WT10G-530 "pheromone" receives much higher score. But for ROBUST04-648 both query terms receive similar scores. We name this category of features as compactness and positions of term score tensors since we intensively compute the all kinds of distances in the multi-dimensional term space and the term scores from the documents are essentially N dimensional vectors. We formally define three types of features in this category as follows:

- Tensor Compactness (**TCP(TC)**): The average and the standard deviation of the distances for the tensors to their centroid. This feature captures the compactness of the tensors cluster.

$$TCP(TC)_{\mu} = \frac{\sum_{d \in L_{q,c}(f)} ||\vec{t_{q,d_i}}(f), \vec{t_{q,d_\mu}}(f)||}{c} \tag{5.15}$$

$$TCP(TC)_{\sigma} = \sqrt{\frac{1}{c} \sum_{d \in L_{q,c}(f)} ||\vec{t_{q,d_i}}(f), TCP(TC)_{\mu}||^2} \tag{5.16}$$

  where f is BM25 ranking model, $||T_A, T_B||$ is the distance between tensor A and tensor B, $\vec{t_{q,d_\mu}}$ is centroid of all the tensors in the list which is essentially

$$\vec{t_{q,d_\mu}}(f) = \left(\frac{\sum \vec{TL_{t_1,c}}(f)}{c}, \frac{\sum \vec{TL_{t_2,c}}(f)}{c}, ... \right) \tag{5.17}$$

- Tensor Closeness to Diagonal (**TCP(CDG)**): The distance from the tensors centroid to the diagonal line in multi-dimensional space, the average and the standard deviation of the distances from the tensors to the diagonal line in multi-dimensional space. These features capture part of the position information of the tensors.

$$TCP(CDG)_c = ||\vec{t_{q,d_\mu}}(f), l_{dg}|| \tag{5.18}$$

$$TCP(CDG)_{\mu} = \frac{\sum_{d \in L_{q,c}(f)} ||\vec{t_{q,d_i}}(f), l_{dg}||}{c} \tag{5.19}$$

$$TCP(CDG)_{\sigma} = \sqrt{\frac{1}{c} \sum_{d \in L_{q,c}(f)} ||\vec{t_{q,d_i}}(f), TCP(CDG)_{\mu}||^2} \tag{5.20}$$

  where l_{dg} is the diagonal line in the multi-dimensional space and $||T, l||$ is the distance from tensor T to line l.

- Tensor Closeness to Nearest Axis (**TCP(CNA)**): We compute the distance from the tensors centroid to its nearest axis, the average/standard deviation distance from the tensors to the nearest axis in multi-dimensional space. TCNA and TCD together define the position property of the tensors.

$$TCP(CNA)_c = ||t_{q,\vec{d_\mu}}(f), l_{na}|| \tag{5.21}$$

$TCP(CNA)_\mu$ and $TCP(CNA)_\sigma$ can be computed similarly with Equation 5.19 and 5.20 with replacement of l_{dg} to l_{na} where l_{na} is the nearest axis to the centroid of all tensors and is computed as:

$$l_{na} = \min_{1 \le i \le N} ||t_{q,\vec{d_\mu}}, l_i|| \tag{5.22}$$

where l_i is ith-axis and N is the number of the dimensions.

We first computed the tensor closeness related features for the terms in the subquery q_i. Later on we found that it is beneficial to compute the tensor closeness related features for all the terms in the original query Q. We apply this in all our experiments.

Basic Features

Besides the aforementioned features (PXM, TS, TCP) we also applied other features proposed by others which we will further refer as "basic features".

Mutual Information (MI)

This feature was proposed by Kumaran et al. [50, 52]. In their original works the MI is computed by first construct a graph for each subquery using the constituent terms as vertexes, and the mutual information between the terms as the weighted edges. "Average" weight is gained after the maximum spanning tree algorithm is applied. We instead propose to compute the MI by first counting the co-occurrence of pairwise terms within N terms window in the matching documents. Then the value is normalized by the product of the DFs of the two terms. Finally we apply all possible feature functions h to the the pairwise terms list. We have

$$MI = h(I(x,y)) = h\left(\frac{\frac{\sum O(x,y)}{T}}{\frac{O(x)}{T} \cdot \frac{O(y)}{T}}\right) \tag{5.23}$$

where $O(x, y)$ is the number of times term x and term y co-occur within a window of 50 terms in each matched documents, $O(t)$ is the total occurrence of term t in the collection and T is the total number of terms in the collection.

Collection Term Frequency (CTF)

The collection level term frequency of term t. Then we apply feature function h to the list of CTFs as $h(CTF_q)$.

Document Frequency (DF)

This is simply the document frequency for each term in the subquery $q_{i,j}$. Then we apply feature function h to the terms DFs as $h(DF_q)$.

Inversed Document Frequency (IDF)

The IDF here is the modified $log(IDF)$ component used in the modified BM25 model [29]:

$$IDF_t = log\frac{N+1}{DF_t}$$

where N is the number of documents in the collection. We then apply the feature function h to the list of IDF_t as $h(IDF_q)$.

Min Document Term Frequency (MINTF) and Max Document Term Frequency (MAXTF)

MINTF is the minimum term frequency in the collection and is computed as:

$$MINTF_t = \min_{1 \leq i \leq DF_t} TF_{t,d_i}$$

Similarly

$$MAXTF_t = \max_{1 \leq i \leq DF_t} TF_{t,d_i}$$

Final features are masked using feature function as $h(MINTF_q)$ and $h(MAXTF_q)$.

Average Document Term Frequency (AVGTF) and Standard Deviation Document Term Frequency (STDTF)

This AVGTF applies to each individual term as:

$$AVGTF_t = \frac{\sum_{i=1}^{DF_t} TF_{t,d_i}}{DF_t}$$

The STDTF is the standard deviation of $AVGTF_t$. We apply feature function masks to both features as $h(AVGTF_q)$ and $h(STDTF_q)$.

Average Document Term Frequency with IDF (AVGTFIDF) and with Collection Occurrence Probability (AVGTFCOP)

Inspired by BM25 model and Dirichlet Language Model we incorporate the average document term frequency with IDF and collection occurrence probability to capture the term salience in the collection. Formally we have:

$$AVGTFIDF_t = AVGTF_t \cdot IDF_t$$

$$AVGTFCOP_t = AVGTF_t + \mu \cdot p(t|C)$$

where $p(t|C)$ is the probability of term t occurred in the whole collection and is computed as $p(t|C) = \frac{CTF_t}{|C|}$. $|C|$ is the total number of terms in the collection. We choose $\mu = 1000$ from the empirical study.

Simplified Clarity Score (SCS)

This feature was firstly proposed by He and Ounis [42] to reduce the computational cost of original query clarity and it was used as a pre-retrieval query performance predicator. It is computed as:

$$SCS_q = \sum_{t \in q} p(t|q) \cdot log_2 \frac{p(t|q)}{p(t|C)}$$

where $p(t|q)$ is the probability of term occurred in the query q.

5.2.3 Experiments and Results

In this section we test our subquery ranking method using TREC collections and topics. We will describe the details of the experiment setup as well as the analysis of the results.

5.2.3.1 Experiment Setup

We use six TREC Ad-hoc/Web collections in our experiments: Disk12, Disk45 with ROBUST04 query set, AQUAINT News Collection with ROBUST05 query set

Table 5.7: Collections and Queries

Collection	#qry	$\|QL\| = 2$	$\|QL\| = 3$	$\|QL\| = 4$
Disk12	150	30(20%)	37(25%)	41(27%)
Disk45	250	75(33%)	147(59%)	17(7%)
AQUAINT	50	21(42%)	27(54%)	1(2%)
WT2G	50	24(48%)	23(46%)	0(0%)
WT10G	100	30(30%)	25(25%)	20(20%)
GOV2	150	44(29%)	65(43%)	35(23%)

(ROBUST04 hard queries), WT2G, WT10G and GOV2. The title part of the query topics is used to test the proposed subquery ranking method. Stopwords are removed from both the collections and the queries and porter stemmer is applied to the indexes. Table 5.7 lists the details of the collections and the corresponding queries. As we can see that for most title topics their lengths are within 2 to 4.

Since we are targeting the subquery ranking, single term queries will not be included [3]. Lots of features can not be directly applied to the queries of length 2 such as MI since the subqueries are single term query (other than the original query) and thus we separate the queries by their lengths. In our experiments we focus on queries of length 2 and 3 since even for queries of length 4 AQUAINT and WT2G do not have enough queries for both training and testing. When tested on one collection, queries from other 5 collections are used together as training examples. All the features are normalized to the range [0, 1] before being fed to the learning algorithm. LambdaMART is leveraged to rank the subqueries based on their features and the average precision of the subquery is used as the labels. Since LambdaMART favors the scalar labels we convert the AP values to integers based on the relative AP values distribution. The relative AP values are the differences between the AP of a subquery and the AP of the best subquery. The mapping from AP to integer should reflect the relative importance of a query whose best subquery performs much better than the rest of its subqueries.

[3] [110] provided the performance upper bound for single term queries

We do not show the actual distribution here but the rule of the mapping is:

$$Label(q_i) = \begin{cases} 4, & \text{if } AP_{q_i} = \max AP_{q_i} \\ 3, & \text{if } AP_{\max q_i} - AP_{q_i} \leq 0.1 \\ 2, & \text{if } AP_{\max q_i} - AP_{q_i} > 0.1 \cap \max AP_{q_i} - AP_{q_i} \leq 0.3 \\ 1, & \text{if } AP_{\max q_i} - AP_{q_i} > 0.3 \cap \max AP_{q_i} - AP_{q_i} \leq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

Basically we only care about which subquery should be labeled as the best-performed subquery thus the metric for LambdaMART is set to nDCG@1. The number of leaves for each tree is chosen from [2, 10] and the best performance averaged among all collections is reported. When generating the ranking list or compute the features like TS and TCP where ranking function is needed we always apply BM25 [4] with optimal parameter b ($k_1 = 1.2$ always) set based on the results reported in [111]. For performance metrics we report the accuracy and MAP. We also compare the MAP of best-performed subqueries identified by our method with the theoretical upper bound.

5.2.3.2 Results of Subquery Ranking

The results of subquery ranking (SR) using all features (mentioned in Section 5.2.2.3 and normalized) for queries of length 2 and 3 are listed in Table 5.8. The accuracy is computed as the number of queries whose best-performed subqueries are correctly identified divided by the total number of queries. MAP is computed by first picking the best subqueries identified by our model and then taking the average precision for these best subqueries. It can be seen that our SR method is better than using original query for most of the collections. The only exception is the GOV2 with queries of length 2 where the upper bound of the optimal performance is almost the same with the performance of using original queries. We also find that in general the our model performs better with queries of length 3 than the queries of length 2 in

[4] We also tried using Dirichlet language model and found using BM25 leads to slightly better performance.

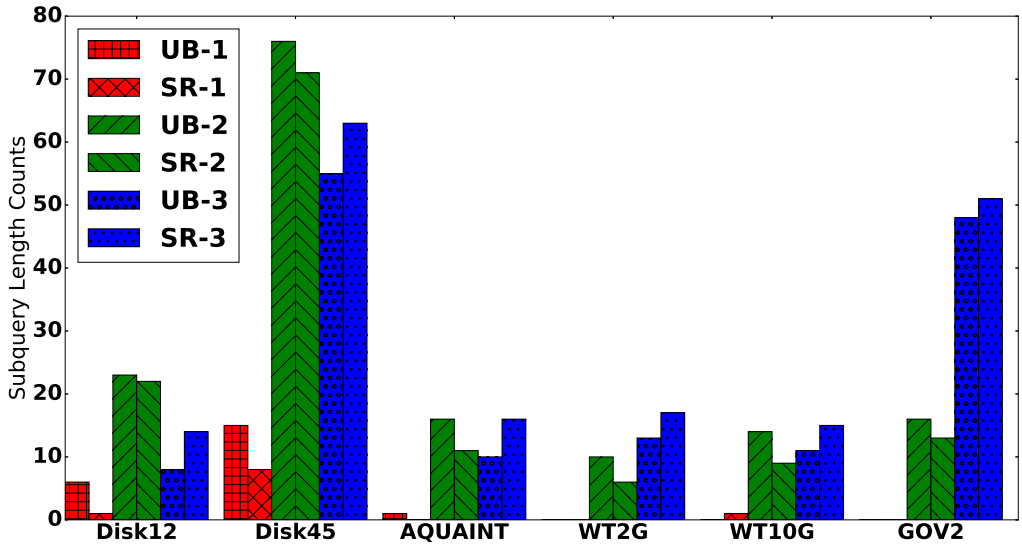
Table 5.8: Results of using all features. *OG* represents the original query. *SR* represents our subquery ranking model. *UB* represents the upper bound where the optimal subquery for each original query is picked.

$\|QL\|$	Collection	Accuracy	MAP		
			OG	SR	UB
2	Disk12	90%	0.3216	0.3309 +2.89%	0.3372 +4.85%
	Disk45	82%	0.2506	0.2566 +2.39%	0.2662 +6.23%
	AQUAINT	76%	0.2063	0.2091 +1.36%	0.2184 +5.87%
	WT2G	83%	0.2983	0.2983 +0.00%	0.3083 +3.35%
	WT10G	83%	0.2544	0.2663 +4.68%	0.2738 +7.63%
	GOV2	96%	0.2912	0.2911 -0.03%	0.2913 +0.03%
3	Disk12	92%	0.2597	0.2833 +9.09%	0.2880 +10.90%
	Disk45	89%	0.2399	0.2643 +10.17%	0.2772 +15.55%
	AQUAINT	88%	0.2107	0.2323 +10.25%	0.2426 +15.14%
	WT2G	90%	0.3285	0.3380 +2.89%	0.3580 +8.98%
	WT10G	94%	0.1720	0.1949 +13.31%	0.2051 +19.24%
	GOV2	95%	0.3060	0.3113 +1.73%	0.3221 +5.26%

terms of percentage improvement. This is mainly because some features such as MI and PXM simply can not be applied to the queries of length 2.

Figure 5.3 shows the length distribution of the best subqueries for the queries of length 3. Basically it shows how many queries of which its best subquery of length N. For example, for Disk12 there are 6 queries whose best subquery are of length 1 and 23 of the queries have their optimal subquery length of 2. From the figure we can see that in general our model has balanced number of best subqueries in different length and the numbers are very close to the upper bound. We also find that our model slightly

Figure 5.3: Optimal Subqueries Lengths of queries with 3 terms. *UB-1* denotes the number of ground truth best subqueries that has 1 term. *SR-1* denotes the number of subquery ranking model ranked best subqueries that has 1 term. *UB-2, UB-3, SR-2, SR-3* follow the same notation.



favors the original queries as the number of best queries that have three terms in our model is always larger than the values of the upper bound.

5.2.3.3 Feature Importance Analysis

In order to quantify the feature importance we set up another experiment in which a subset of features are taken off from the the feature space and the performance difference between using all features and using the subset of features is compared. The results are in Table 5.9 and we only show the results of queries of length 3.

In Table 5.9 there are two main sections: the top important features from basic features are on the left and the detailed feature importance of terms relationship features on the right. For basic features the features with the largest performance drops if they were removed from the feature space are listed. For terms relationship features we present the details of PXM, TS and TCP by showing the importance of the sub-features. Sub-features are the features like $PXM(w)_h$ and $TCP(CDG)$ which essentially reflect

specific intuitions of the newly proposed features. First, we notice that in general the terms relationship features have the performance drop larger than -13%. Comparing to the top basic features where two of them have the performance drop below than -13% it validates the utility of these features. Second, Detailed collection-wise performance drop indicates different features contribute differently for the collections. For example, AVGTFCOP is important to AQUAINT while TS(SUM,SUM) is specifically useful for WT2G. Third, detailed analysis on terms relationship features reveal that: for PXM $PXM(w)_h$ is better than $PXM(w)_corr$. For TS features TS(SUM,SUM) which is actually the sum of the top ranked documents scores (the first sum of all query terms equals to the document score) is more vital. For TCP features the TCP(CNA) which captures the position of the terms scores tensors and the TCP(TC) which captures the tensors compactness are all important with performances drops larger than -14% and this validates the utility of such features.

5.3 Summary and Future Work

We have discussed about the tools to analyze the traditional ranking models – their performance upper bound and the possible best subqueries for keyword query. We find that classic ranking models such as BM25 and Dirichlet language model are quite close to the practical performance upper bound for single-term queries. We further notice that for keyword queries the best subqueries can achieve better performance. By exploring the post-retrieval features like term proximity, term score and scores tensor properties we can reach the projected performance.

As for future work, there are two interesting directions. The first one is to continue on the features so that the identification of the best subquery can be further improved. For example, the semantic features would be the promising perspective. Incorporating the outside resources as the potential feature source would be another choice. The second direction is to leverage the terms relationship features and the experimental results presented in this thesis to do more theoretical studies. For example,

we could get inspiration from the features and try to prove the performance upper bound of multiple-terms queries.

Table 5.9: Feature importance analysis for queries of length 3, the lower the better. The lowest value of each collection is bolded. **TS1**: TS(MAX/MIN,SUM); **TS2**: TS(SUM,SUM); **TS3**: TS(GMEAN,MEAN)

(a) Top Basic Features

Collection	Top Basic Features				
	AVGTFCOP	SCS	CTF	$PXM(w)_h$	$PXM(w)_{corr}$
Disk12	0.2399	0.2497	0.2445	0.2535	0.2497
0.2833	-15.3%	-11.9%	-13.7%	-10.5%	-11.9%
Disk45	0.2292	0.2293	**0.2224**	0.2306	0.2354
0.2643	-13.3%	-13.2%	**-15.9%**	-12.8%	-10.9%
AQUAINT	**0.1882**	0.1999	0.2003	0.1974	0.1970
0.2323	**-19.0%**	-13.9%	-13.8%	-15.0%	-15.2%
WT2G	0.2561	0.2756	0.2853	0.2785	0.2760
0.3380	-24.2%	-18.5%	-15.6%	-17.6%	-16.3%
WT10G	0.1587	0.1643	0.1663	0.1454	0.1510
0.1949	-18.6%	-15.7%	-14.7%	-25.4%	-22.5%
GOV2	0.3040	0.2990	0.3029	0.2989	0.3025
0.3113	-2.3%	-4.0%	-2.7%	-4.0%	-2.8%
AVG	0.2294	0.2363	0.2370	0.2341	0.2364
0.2707	-15.5%	-12.9%	-12.7%	-14.2%	-13.3%

(b) TS and TCP Features

Collection	Features					
	TS1	TS2	TS3	TCP(TC)	TCP(CDG)	TCP(CNA)
Disk12	0.2532	0.2536	0.2536	**0.2374**	0.2492	0.2498
0.2833	-10.6%	-10.5%	-10.5%	**-16.2%**	-12.0%	-11.8%
Disk45	0.2294	0.2313	0.2313	0.2330	0.2337	0.2267
0.2643	-13.2%	-12.5%	-12.5%	-11.8%	-11.6%	-14.2%
AQUAINT	0.2029	0.1949	0.1949	0.1884	0.2005	0.1999
0.2323	-12.7%	-16.1%	-16.1%	-18.9%	-13.7%	-13.9%
WT2G	0.2641	**0.2191**	0.2191	0.2828	0.2826	0.2795
0.3380	-21.9%	**-35.2%**	-35.2%	-16.4%	-17.3%	-18.3%
WT10G	0.1671	0.1617	0.1617	0.1589	**0.1435**	0.1441
0.1949	-14.3%	-17.0%	-17.0%	-18.5%	**-26.4%**	-26.1%
GOV2	**0.2857**	0.2927	0.2947	0.3046	0.3087	0.2990
0.3113	**-8.2%**	-6.0%	-5.3%	-2.2%	-0.8%	-4.0%
AVG	0.2337	**0.2256**	0.2259	0.2342	0.2359	0.2329
0.2707	-13.5%	**-16.2%**	-16.1%	-14.0%	-13.6%	-14.7%

Chapter 6

CONTEXTUAL SUGGESTION

The increasing use of mobile devices enables an information retrieval (IR) system to capitalize on various types of contexts (e.g., temporal and geographical information) about its users. Combined with the user preference history recorded in the system, a better understanding of users' information need can be achieved and it thus leads to improved user satisfaction. More importantly, such a system could *proactively* recommend suggestions based on the contexts.

In this chapter, we first introduce our effort on developing the context tracking mobile application called UDTracker. UDTracker can silently track the user's location and the current time which enables the precise capture of the context of the users.

User profiling is essential and is the key to success in contextual suggestion. Since user's preference is always modeled as long-term and static we include it as part of the context of this problem. Given most users' observed behaviors are sparse and their preferences are latent in an IR system, constructing accurate user profiles is generally difficult. In our work, we focus on location-based contextual suggestion and propose two approaches to construct user profiles.

The first approach uses the categories and/or descriptions from users' activities history to build user profile. The rationale here is that users are at a better chance to favor the places that are similar to what she liked before in terms of the category/description of the places. In reality, one user would typically have several positively rated and also several negatively rated suggestions in the past. We compute the similarity of category/description between each candidate suggestion and all places in the user's activity history and combine the averages of both positive and negative scores.

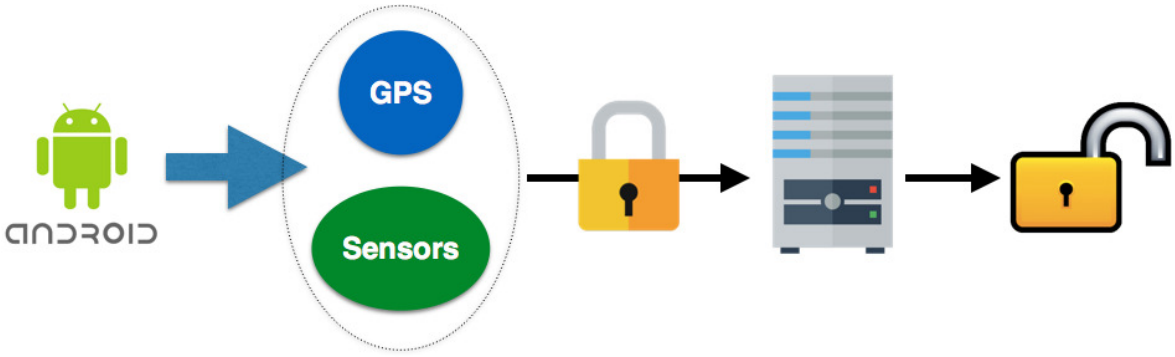


Figure 6.1: Data is immediately encrypted and uploaded. Encryption uses AES, RSA with salt.

The second approach leverages the users' opinions to form the user profiles. By assuming users would like or dislike a place with similar reasons, we construct the opinion-based user profile in a collaborative way: opinions from the other users are leveraged to estimate a profile for the target user. Candidate suggestions are represented in the same fashion and ranked based on their similarities with respect to the user profiles.

Moreover, we also develop a novel summary generation method that utilizes the opinion-based user profiles to generate personalized and high-quality summaries for the suggestions.

Experiments conducted over three standard TREC Contextual suggestion collections and a Yelp data set show the advantage of our approaches and the system developed based on the proposed methods have been ranked as top 1 in both TREC 2013 and 2014 Contextual Suggestion tracks.

6.1 Mobile Context Tracking Application

We first introduce our effort of developing the mobile context tracking application called UDTracker. UDTracker is an Android application which requires user's permission to passively collect the information such as geo-location, time from users. The main concern of the application, however, lies in the security of the application due

to the sensitivity of the data we collect from the user. The strategy is to directly encrypt the collected data on user's phone and then transfer the encrypted and archived data to the secure server where the data is uncompressed and decrypted. After the data is decrypted we can do our analysis of the data. The design phlogsoly can be found in Figure 6.1.

6.2 Problem Formulation of Contextual Suggestion

The problem of contextual suggestion can be formalized as follows. Given a user's contexts (e.g., location and time) and the her/his preferences on a few example suggestions, the goal is to retrieve candidate suggestions that can satisfy the user's information need based on both the context and preferences. For each returned candidate suggestion, a short description may also be returned so that the user could decide whether the suggestion is interesting without going to its website. For example, assume that a user liked "Magic Kingdom Park" and "Animal Kingdom", but disliked "Kennedy Space Center". If the user is visiting Philadelphia on a Saturday, the system is expected to return a list of suggestions such as "Sesame Palace" together with a short summary of each suggestion, e.g., "Sesame Place is a theme park in Langhorne, Pennsylvania based on the Sesame Street television program. It includes a variety of rides, shows, and water attractions suited to very young children."

Since our paper focuses on user modeling, we assume that we have filtered out the suggestions that do not meet the context requirement and the remaining suggestions only need to be ranked based on the relevance to user preferences. Note that the filtering process based on contexts can be achieved by simply removing the suggestions that do not satisfy the contextual requirements, such as the ones that are either too far away from the current location or those that are currently closed.

The remaining problem is essentially a ranking problem, where candidate suggestions need to be ranked based on how relevant the suggestions are with respect to a user's interest. Formally, let U denote a user and CS denote a candidate suggestion, we need to estimate $S(U, CS)$, i.e., the relevance score between the user and the

suggestion.

It is clear that the estimation of the relevance score is related to how to represent U and CS based on the available information. Let us first look at what kind of information we can gather for U and CS. For each user U, we know the user's preferences (i.e., ratings) for a list of example suggestions. We denote an example suggestion ES and its rating given by user U as $R(U, ES)$. For a suggestion (either CS or ES), we assume that the following information about the suggestion is available: the text description such as title and category and online opinions about this suggestion. Note all the information can be collected from online location services such as Yelp and Tripadvisor.

6.3 Category and Description based User Profile Modeling

6.3.1 Ranking Based on User Profiles

We first describe our framework of how to rank candidate suggestions based on user profiles. How to use the category and description to build user profile will be introduced later. The profile of each user consists of the user's preferences for example suggestions. The suggestions that a user likes are referred to as "positive user profile", and those disliked by the user are referred to as "negative user profile". Intuitively, the relevance score of a candidate suggestion should be higher when it is similar to positive user profile while different from the negative user profile.

Formally, we denote U as a user and CS as a candidate suggestion. Moreover, let $\mathcal{U}_+(U)$ denote positive user profile, i.e., a set of places that the user likes, and $\mathcal{U}_-(U)$ denote negative user profile, i.e., a set of places that the user dislikes. The relevance score of CS with respect to U can then be computed as follows:

$$\begin{aligned} \mathrm{S(U, CS)} &= \varphi \times \mathrm{SIM}(\mathcal{U}_+(U), CS) + (1-\varphi) \times \mathrm{SIM}(\mathcal{U}_-(U), CS) && (6.1) \\ &= \varphi \times \frac{\sum_{e \in \mathcal{U}_+(U)} \mathrm{SIM}(e, CS)}{|\mathcal{U}_+(U)|} + (1-\varphi) \times \frac{\sum_{e \in \mathcal{U}_+(U)} \mathrm{SIM}(e, CS)}{|\mathcal{U}_+(U)|} && (6.2) \end{aligned}$$

where $\varphi \in [0, 1]$ regularizes the weight between the positive and negative similarities. When $\varphi = 1$, the highly ranked suggestions would be those similar to the suggestions

Table 6.1: Examples of Categories in Example Suggestions

NAME	**Categories**
HoSu Bistro	SushiRestaurant→Restaurants; KoreanRestaurant→Restaurants; JapaneseRestaurant→Restaurants
The Rex	JazzBlues→MusicVenues→Arts
St. Lawrence Market	Grocery→Shopping; FarmersMarket→Shopping
...	...

that the user likes. When $\varphi = 0$, the highly ranked suggestions would be those different from the suggestions that the user dislikes. SIM($\mathcal{U}_+(U), CS$) measures the similarity between the positive user profile and the candidate suggestion, and we assume that it can be computed by averaging the similarity scores between each positive example and the candidate suggestion. $|\mathcal{U}_+(U)|$ corresponds to the number of positive examples in the user profile. It is trivial to explain the corresponding symbols for negative one.

Thus, it is clear that the problem of computing the relevance score of a candidate suggestion with respect to a user can be boiled down to the problem of computing the relevance score between a candidate suggestion and a place mentioned in the user profile, i.e., SIM(e, CS), where e is an example from the user profile. We explore the following two types of information to compute SIM(e, CS): (1) the category of a place; and (2) the description of a place.

6.3.1.1 Category-based Similarity

Category is a very important factor that may greatly impact user preferences. However, the categories of the suggestions are often in hierarchical format. Here is an example category, *[History Museum→Museum→Arts].* The categories becomes more general from the left to the right. In this example, *Arts* is the most general category while *History Museum* is the most specific category. Note that we represent the hierarchical categories as a set of categories in this paper.

We can compute $\text{SIM}(e, CS)$ based on the category similarities between e and CS as follows:

$$\text{SIM}_{\mathcal{C}}(e, CS) = \frac{\sum_{c_i \in \mathcal{C}(e)} \sum_{c_j \in \mathcal{C}(C)} \frac{|Intersection(c_i, c_j)|}{max(|c_i|, |c_j|)}}{|\mathcal{C}(e)| \times |\mathcal{C}(C)|} \tag{6.3}$$

where $\mathcal{C}(e)$ denotes the set of categories of location e and $|Intersection(c_i, c_j)|$ is the number of common categories between c_i and c_j. Recall that we crawled the candidate suggestions from two online sources. Table 6.1 shows some example categories of the suggestions.

6.3.1.2 Description-based Similarity

In example suggestions, each suggestion has its unique description which typically is at a short length. We want to learn how the descriptions can affect people's decision on different places. By comparing the descriptions of training suggestions with textual web sites of testing suggestions we may find some interesting connections. We use textual web sites of testing suggestions because we believe that textual web sites are more reliable than descriptions especially when we rank candidate suggestions. The similarity used function is the F2EXP ranking function [33] since it has been shown to be effective for long queries [33]. So, we have

$$\text{F2EXP}(a, b) = \sum_{t \in a \cap b} \frac{c(t, b)}{c(t, b) + 0.5 + \frac{0.5 \cdot |b|}{avdl} \cdot (\frac{N+1}{df(t))})^{0.35}} \tag{6.4}$$

Thus, we compute the similarity scores as follows:

$$\text{SIM}_{\mathcal{D}}(e, CS) = \text{F2EXP}(DES(e), DES(CS)) \tag{6.5}$$

where $\text{DES}(e)$ is a description of the example place e.

6.4 Opinion-based User Profile Modeling

6.4.1 Basic Idea

In our problem setup, the available information for a user U includes the user's preferences for a set of example suggestions. Existing studies often estimated user

profiles based on the descriptive information of the example suggestions such as their names, descriptions and web sites [7, 46, 49, 85, 105–107]. However, one limitation of this approach is that such descriptive information could be very specific for one suggestion and might not be useful at all to infer the user's preferences on other suggestions. Categories of the suggestions were then used by some methods to overcome the limitation [88, 105, 115]. Although this method improves the performance, the improvement is often limited since category information might be too general to capture the reasons behind the user preferences.

Instead of simply capturing what a user likes or dislikes, i.e. the descriptive information of example suggestions, we propose to model the user profile based on the user's opinions about the example suggestions. The opinions about a suggestion is defined as the ⟨ rating, review text ⟩ pairs in our paper. When determining whether an opinion is positive or negative, we rely on the numeric rating rather than the review text. More details about this are described in Section 6.6.2.1.

We now motivate the opinion-based user modeling through an example as shown in Figure 6.2. Assume that we know a user's preferences for the first four suggestions and want to infer the user preference for the last one. Neither description-based nor category-based methods are effective here. For example, the category of the candidate suggestion is "hotel", which does not match with the categories of all the example suggestions. Moreover, the descriptions of these example suggestions are very specific, making it difficult to find their commonalities. However, if we are able to know the user's preference and review for each example suggestion, it would be possible for us to more accurately infer why the user liked or disliked these places. For example, it seems that the two suggestions that the user liked (i.e., example suggestions 1 and 3) are "clean" while the places that the user disliked (i.e., example suggestions 2 and 4) are both "dirty". Thus, we may infer that the user prefers places that are "clean". Now if we know that a candidate suggestion is well known for its "cleanness" based on online reviews, we could safely infer that the user would like this candidate suggestion. Clearly, opinion- based user profile modeling should be more effective than the category- based

and description-based methods since it can capture user preferences more accurately.

One challenge of using opinions to model user profile is that users may not share their opinions explicitly by writing the reviews for each example suggestion. To address the challenge, we propose to leverage opinions from similar users. More specifically, we assume that users who rate a suggestion similarly would share the similar opinions about the suggestion. If a user likes a suggestion, we could identify all other users who also like this suggestion and leverage their reviews about the suggestion as part of the user's positive profile, i.e., the profile about what the user likes. We can build the negative profile in a similar way.

Specifically, we use positive reviews of the example suggestions that the user likes to build his or her positive user profile, and use negative reviews of example suggestions that the user dislikes to build negative user profile. The basic assumption is that the opinion of a user about a place can be inferred by the opinions of the users who share the same preference as the target user to the same place.

Formally, a user U 's positive profile $\mathcal{U}_+(U)$ can be estimated as follows:

$$\mathcal{U}_+(U) = \bigcup_{\forall i, R(U,ES_i)=POS} REP_+(ES_i), \tag{6.6}$$

where ES_i is an example suggestion and $R(U, ES_i)$ is the rating of ES_i given by user U. The ratings could be binary or within a specified range, but they can be mapped to either positive (i.e., POS) or negative (i.e., NEG). We will provide more details on these mappings in our experiment setup. $REP_+(ES_i)$ is the positive opinion based representation for ES_i and we will provide more details about the representation in the following subsection (i.e., Section 6.4.2).

Similarly, a user U's negative profile $\mathcal{U}_-(U)$ can be estimated as:

$$\mathcal{U}_-(U) = \bigcup_{\forall i, R(U,ES_i)=NEG} REP_-(ES_i), \tag{6.7}$$

where $REP_-(ES_i)$ is the negative opinion based representation for ES_i.

6.4.2 Opinion-based Representation for Suggestions

We now discuss how to generate opinion-based representations for the suggestions (CS or ES). Given an ES, we need to construct two profiles: (1) positive profile, i.e., $REP_{+}(ES)$, based on all the positive reviews of ES; and (2) negative profile, i.e., $REP\_(ES)$ based on all negative reviews of ES.

Now the remaining challenge is how to construct these two profiles based on the reviews. For example, do we include every term from the reviews? Or shall we only include important terms from the reviews? If so, how to select the important terms and what are the impact of the selected terms? In order to answer all these questions, we explore the following four strategies to construct $REP_{+}(ES)$ and $REP\_(ES)$ based on the reviews. All of these strategies are based on "bag-of-terms" representations but they are different in which terms from the reviews are used in the representations.

- *Full reviews (**FR**)*: The simplest approach is to take all terms occurring in the review text to build the profile. For example, when estimating $REP_{+}(ES)$, we take all the positive reviews about ES and use bag of terms representations for these reviews. We can estimate $REP\_(ES)$ in a similar way using negative reviews. Despite its simplicity, this representation may cause the efficiency concern because when more reviews are available, the size of the profiles could be fairly large.

- *Selective term based reviews (**SR**)*: To reduce the computational cost, one possibility would be to construct the profile based on a set of selected terms. Terms could be selected using different criteria, and we include the most frequent terms in the profiles. Specifically, top 100 most frequent terms in the review text are selected and their frequencies are set to 1 after being selected. This strategy would be less computational expensive than the FR method, but it may not perform as well since using only frequent terms might not be the best way of representing opinions.

- *Noun based reviews (**NR**)*: Another strategy that we have explored to generate concise profiles based on reviews is to only use the nouns from the review text. The rationale is that nouns often correspond to important aspects of a suggestion, and nouns are less noisy than the frequent terms. Thus, we expect better performance of this method compared with **SR**.

- *Review summaries (**RS**)*: Finally, we leverage the Opinosis algorithm [37], an unsupervised method that generates concise summaries of reviews, to construct the profiles. The algorithm first generates a textual word graph (called the

Opinosis-Graph) of the input data, where each node represents a word, and an edge represents the link between two words. Using three unique properties of the graph data structure (redundancy capture, collapsible structures, gapped sub-sequences), various promising sub-paths in the graph that act as candidate summaries are scored and ranked. The top candidate summaries are then used to generate the final Opinosis summaries. In this work, we first concatenate all the reviews and then generate the review summary using the Opinosis algorithm.

Figure 6.3 shows an example of the original review and the results of different opinion-based representations. When building user profile models, we perform the following simple pre-processing on the original reviews: 1) converting terms into lower cases; and 2) removing punctuations and stop words.

6.4.3 Candidate Suggestions Ranking

We now describe how to rank candidate suggestions based on the user profiles. As described in the previous section, we can estimate a user's profile based on the user's preferences on the example suggestions as well as the reviews of the example suggestions. In particular, the profile of user U can be represented with $\mathcal{U}_+(U)$ and $\mathcal{U}_-(U)$. Similarly, a candidate suggestion CS can be represented based on its positive and negative reviews, i.e., $REP_+(CS)$ and $REP_-(CS)$. Thus, the relevance score $S(U, CS)$ should be related to the similarities between the positive/negative user profiles and the positive/negative representations of candidate suggestions.

In order to compute $S(U, CS)$, we investigate two possible ways of combining these similarities: linear interpolation and learning-to-rank.

6.4.3.1 Linear Interpolation

Linear interpolation is a simple yet effective method to combine multiple scores into one. The main idea here is to linearly combine the similarity scores between user profiles (i.e., $\mathcal{U}_+(U)$, $\mathcal{U}_-(U)$) and the candidate profiles (i.e., $REP_+(CS)$ and $REP_-(CS)$).

In the previous section, we have discussed how to construct these profiles, now we discuss how to compute their similarities. Our basic idea is illustrated in Figure 6.4. Intuitively, a user would prefer suggestions with the properties that the user likes or those without the properties that the user dislikes. This means that the relevance score $S(U, CS)$ should be positively correlated with the similarity between two positive profiles and two negative profiles, i.e., $\text{SIM}(\mathcal{U}_+(U), REP_+(CS))$ and $\text{SIM}(\mathcal{U}_-(U), REP_-(CS))$. Similarly, a user would not like suggestions with the properties that the user dislikes or suggestions without the properties that the user likes, which means $S(U, CS)$ should be negatively correlated with the similarity between positive and negative profiles, i.e., $\text{SIM}(\mathcal{U}_+(U), REP_-(CS))$ and $\text{SIM}(\mathcal{U}_-(U), REP_+(CS))$.

Following the above intuitions, we can estimate the similarity between a user and a candidate suggestion as follows:

$$\begin{aligned} S(U, CS) = \alpha \times \text{SIM}(\mathcal{U}_+(U), REP_+(CS)) - \beta \times \text{SIM}(\mathcal{U}_+(U), REP_-(CS)) \\ - \gamma \times \text{SIM}(\mathcal{U}_-(U), REP_+(CS)) + \eta \times \text{SIM}(\mathcal{U}_-(U), REP_-(CS)) \end{aligned} \tag{6.8}$$

where α, β, γ and η are parameters that balance the impact of the four components to the final similarity score. All of their values are between 0 and 1. $SIM(a, b)$ could be any text similarity measure. In this paper, we used an axiomatic retrieval function F2EXP [33] since it has been shown to be effective for long queries [33]. So, we have

$$\text{SIM}(a, b) = \sum_{t \in a \cap b} \frac{c(t, b)}{c(t, b) + 0.5 + \frac{0.5 \cdot |b|}{avdl} \cdot (\frac{N+1}{df(t)})^{0.35}} \tag{6.9}$$

where $c(t, b)$ is the occurrences of term t in b and $|b|$ is the number of terms in b. $avdl$ is the average length of all the candidate suggestion representations, N is the number of candidate suggestions in the collection, and $df(t)$ is the number of candidate suggestion representations that contain term t. Note that there are two collections for the candidate suggestion representations, i.e, positive one vs. negative one. Depending on whether b is a positive or negative representation, the last three statistics are computed based on the corresponding collection.

6.4.3.2 Learning to Rank

Machine learning is another way of combining multiple features. And learning to rank has been proven to be effective in information retrieval area [65, 70].

For our task, we can first compute the similarity scores
$\text{SIM}(\mathcal{U}_+(U), REP_+(CS))$, $\text{SIM}(\mathcal{U}_-(U), REP_-(CS))$,
$\text{SIM}(\mathcal{U}_+(U), REP_-(CS))$ and $\text{SIM}(\mathcal{U}_-(U), REP_+(CS))$ which is exactly the same as what we do in linear interpolation method (Section 6.4.3.1). After having these similarities at hand, we can use the similarities as features and use learning-to-rank methods to compute the ranking score for each candidate suggestion. The following learning-to-rank methods are considered:

- **MART**, which is also known as Gradient Boosted Regression Trees. It generates a set of weighted regression trees that aim to predict the scores of training data [36]. The regression tree learned at each iteration only needs to focus on the difference between the target label and the prediction of previous trees. The number of trees can be tuned via the validation data.
- **LambdaMART**, which also applies boosted regression trees, but the training of the trees consider numeric measurements (such as NDCG and ERR) to obtain the gradient of the surrogate loss function between pairs of documents [9]. Like MART, the number of iterations can also be tuned via the validation data. It is denoted as LMART in the paper.
- **LinearRegression**, which views the target label as a linear combination of the attributes. The goal is to search for parameters so that the sum of the squares of differences between target label and the predicted label is minimized. It is denoted as LR in the paper.

6.5 Structured Summary Generation

Here we discuss how to generate a *personalized* and *structured* summary for a candidate suggestion. A straightforward solution is to apply existing text summarization techniques and extract important information from the website of a suggestion [25]. The result would be similar to the search snippets generated by Web search engines for the suggestion's website. For example, the snippet of *Olive Room* [1] is "The Olive Room, French Restaurant in Montgomery. See the menu, 49 photos, 1 blog post and 34 user reviews. Reviews from critics, food blogs and fellow ..."

[1] http://www.theoliveroom.com

Although this strategy would work, it might not be optimal for the following reasons. First, the summary comes from only a single information source, i.e., the website of the suggestion, which may lead to incomplete or even biased information about the suggestion. Second, the summary is not personalized. The lack of personalization might not effectively convince every user.

To overcome these limitations, we propose a novel summarization method for contextual suggestions that leverages the user profile as well as the information from multiple sources about the suggestions to produce *personalized* and *structured* summaries.

Given a suggestion, we could collect a wide variety of information about the suggestion, which includes the category of the suggestion, website of the suggestion as well as the reviews of the suggestion. Note that the category and reviews of a suggestion can be downloaded from the third party websites such as Yelp and Tripadvisor. Recall that the user profiles we have estimated can tell us what makes a user like or dislike a suggestion. Thus, it would be interesting to study how to leverage user profiles to generate summaries that are more convincing. Now, the key challenge is how to synthesize the information from various sources and generate a coherent personalized summary.

To tackle this problem, we propose to generate a structured summary. In particular, the summary consists of multiple fields, and each field aims to provide information about a unique aspect of the suggestion. All the fields together would offer a more complete information about the suggestion as well as arguments on why the suggestion would be appealing to a particular user.

The structured summary consists of the following four components:

- **An Opening Sentence:** It provides a high-level introduction in one sentence.
- **An "official" introduction:** It provides more detailed information about the suggestion by extracting information from the website of the suggestion.
- **Highlighted reviews:** This component explains why other people like this suggestion based on the information extracted from the reviews.

- **A concluding sentence:** This component explains why this suggestion is recommended to the user.

We now provide more detailed information on how to generate the above structured summary.

An opening sentence. The opening sentence serves as a high-level introduction sentence. Sometimes people can even hardly know what kind of the suggestion it is by looking at its name. For instance, we might guess that "Zahav" is related to food, but what kind of food? Intuitively, the opening sentence should clearly explain what this suggestion is. And the category information of this suggestion could be a good choice. Our opening sentence then is of the form: suggestion's name followed by the fine category of that suggestion. For example, "The big fish grocery is a *shopping store especially for seafood*." If the fine category of candidate suggestion is not available, we show its coarse category like "The DCM is a museum." The fine and coarse category can be obtained from the data sources such as Yelp and Google Place.

The "official" introduction. The "official" introduction consists of useful sentences extracted from the web site of the suggestion. Generally speaking, we cannot rely on the HTML DOM structure to extract the well crafted description for two reasons: (1) there might not be dedicated field to store such information, even in the meta data; (2) even if we can find a short summary in the meta data, the information might be too general and does not match user interests well. To address this challenge, we propose to leverage reviews to identify important information from the websites. Specifically, we first extract nouns with high frequency from the suggestion opinions. After that, we use these nouns to identify the sentences from the web site of the candidate suggestion. All the identified sentences are ranked based on the number of distinctive/total positive adjectives. Only top 5 ranked sentences are used due to the length of the summary.

The highlighted reviews. The highlighted reviews are the sentences extracted from the positive reviews of the suggestion. The process is very similar with the extraction of "official" introduction. We use the most frequent nouns as a guide to extract

sentences from positive reviews. Sentences with more distinct positive adjectives are chosen.

The concluding sentence. The concluding sentence is the last sentence in the structured description. Here we customize it to specific user. The concluding sentence is of the form: "We recommend this suggestion to you because you liked *abc* and *xyz* in example suggestions." *abc* and *xyz* are example suggestions that have the same fine category as the candidate suggestion.

As an example, here is the generated summary for a candidate suggestion, i.e., *Olive Room. "The Olive Room is a bar. HERE ARE THE DESCRIPTIONS FROM ITS WEBSITE: Here at the olive room, you will receive the finest cuisine montgomery has to offer, hands down. HERE ARE REVIEWS FROM OTHER PEOPLE: If you are looking for a unique dining experience, with excellent food, service, location, and outstanding ambiance, look no further! THIS PLACE IS SIMILAR TO OTHER PLACE(S) YOU LIKED, i.e. Tria Wine Room."*

6.6 Experiments

We conduct experiments to evaluate the proposed category-based, description-based and opinion-based candidate ranking methods as well as the summarization method.

6.6.1 Data sets

To evaluate the effectiveness of the proposed methods, we conduct experiments over two types of data sets: (1) the data set used in the TREC Contextual Suggestion track [27]; and (2) a data set crawled from Yelp[2].

- **TREC data set:** The TREC Contextual Suggestion Track [27] provides an evaluation platform for the problem of contextual suggestion. We use the officially released collections from 2012 to 2014, and denote them as *CS2012*, *CS2013* and *CS2014* respectively. Each collection consists of a set of example suggestions and user profiles. User profile includes the ratings for each suggestion given by each

[2] http://www.yelp.com

Table 6.2: Statistics of the three TREC collections

Collection	Num. of Users	Num. of Suggestions	the range of ratings
CS2012	34	49	[-1,1]
CS2013	562	50	[0,4]
CS2014	299	100	[0,4]

user. The information provided about each example suggestion includes its name, a short description and the URL to its webpage. To gather the opinions for each suggestion, we crawl the ratings and text reviews of the suggestions from Yelp. The statistics of these three TREC collections are summarized in Table 6.2.

- **Yelp data set:**[3] For the TREC collections, all users rated the same number of suggestions, which might not be the case in reality, e.g., sparse observations of users' preferences. To assess the proposed methods in a more realistic setting, we construct another evaluation data set based on Yelp reviews. Specifically, we randomly picked 100 Yelp users, and crawled the information about suggestions they had rated as example suggestions in one month period (from January 15, 2013 to February 14, 2013). Note that, for each suggestion, we have its name but do not have the short description as in the TREC collection. The total number of crawled suggestions is 13,880. All the opinions (i.e., ratings and text reviews) about each suggestion are also crawled. The users ratings are in the range of [1,5].

These two evaluation data sets have distinct characteristics. In the TREC collections, there is a fixed set of example suggestions, and all the users provide their ratings on those suggestions. On the contrary, in the Yelp collection, different users would rate different sets of suggestions, where the overlapped suggestions are small and the number of rated suggestions per user also varies. The average number of rated suggestions per user is around 200.

6.6.2 Experiments on Candidate Suggestion Ranking

6.6.2.1 Experiment Design

In all the collections, for each user, we need to split the suggestions that rated by this user into development set and test set. The suggestions in the development set are used to construct user profile while those in the test set are used as candidate

[3] available at https://s3.amazonaws.com/irj2014_yelp_data/irj2014_yelp.tar.gz

suggestions that need to be ranked. For each user, we randomly select 50% of the suggestions from each category at each rating level to build the user profile, and use the remaining ones as the test set. We will discuss the impact of the size of development set for user profile construction in Section 6.6.2.3.

As discussed in Section 6.6.1, user rating values in different evaluation collections are different. We need to map them into either POS (i.e, positive) or NEG (i.e., negative) as described in Equation 6.6. In the *CS2012* data set, the rating of 1 is mapped to POS and the ratings of -1,0 are mapped to NEG. In the *CS2013* and *CS2014* data sets, the ratings higher than 2 are mapped to POS while those lower than 2 are mapped to NEG. In the *Yelp* data set, the ratings higher than 3 are mapped to POS while those lower than 3 are mapped to NEG. Note that the reviews assigned with the middle rating are not included in the mapping because it is difficult to directly classify them into positive or negative opinions without looking at the text reviews.

The evaluation measures for candidate suggestion rankings are P@5 (precision at top 5 results) and ERR@20 (expected reciprocal rank at top 20 results) [17]. P@5 is the official measure used in the TREC contextual suggestion track. Since the relevance judgement of a candidate suggestion is graded and P@5 cannot capture the graded relevance, we use ERR@20 as an additional measure.

6.6.2.2 Results of candidate suggestion ranking

We first conduct experiments to evaluate the proposed methods when using linear interpolation. The results of using 5-fold cross validation are shown in Table 6.3. The Yelp data set does not have description for each suggestion to build the user profile, so the description-based method is not applicable for this data set.

We can see that opinion-based methods have better performances over category-based or description-based methods over both measures and all the collections. These results show that it is more effective to model user preferences using the opinions about the suggestions than using the categories or the descriptions of the suggestions. In particular, the improvement is larger on the Yelp data collection. This indicates that

the opinion-based methods can capture the user preferences in a more general way. Moreover, the evaluation results of all the opinion-based methods are quite similar; among them, NR seems to be the most stable one.

There are four parameters in the linear interpolation methods as described in Section 6.4.3. We find that the optimal parameter setting is as follows: $\alpha = 1.0, \beta = 0.0, \gamma = 0.9, \eta = 0.1$, which indicates both positive and negative user profiles are important. It verifies our hypothesis that it is necessary to capture both what a user likes and what a user dislikes in contextual suggestion. Furthermore, we can find that the positive candidate suggestion representation is more useful than the negative one.

Table 6.4 shows the performance of learning-to-rank methods. All the models are trained on 60% of the data, validated on 20% of the data, and then tested on the remaining data. This process is repeated 5 times and the average performance is reported. We can see that the opinion-based user profiling is still consistently better than the description or category-based methods. Among the three learning-to-rank methods, LMART and MART performed much better than the linear regression methods, and MART was the best. Among different representations, the performance is still similar, and NR remains to be a reasonable choice.

Based on the results of these two tables, it seems that the best strategy is to use NR for opinion-based representation and use MART to combine the similarities.

6.6.2.3 In-depth Analysis

We first conduct experiments to analyze how the size of development set used to build the user profile affects the performance of these methods. In the previous experiments, for each user, we used 50% of the suggestions rated by the user to build user profiles. It is necessary to verify how the performance changes when fewer suggestions are used. The results are shown in Figure 6.5. The X axis indicates the percentage of suggestions used to build the profile, and the Y axis corresponds to the ranking performance. It is clear that the performance of the opinion-based method (i.e., NR)

is more robust with respect to the quality of the user profile. Even when we use fewer number of suggestions to build the profile, the performance remains robust.

Previous results show that NR seems to be more robust and effective than the other profile representations. Our result analysis suggests that the better performance may be related to the fact that the NR-based profiles contain fewer noisy terms. Here, we use an example pair i.e., user (uid:918) and candidate suggestion (id:107), to illustrate it. Table 6.5 shows the most frequent terms in the positive user profiles and the positive representation of the candidate suggestion. We can see that the candidate is about a place that selling "breakfast items" while the user seems to like "beers" and "chicken wings". Comparing these different profiles, it is clear that the profiles generated by NR contain fewer noisy terms than others. When computing the similarity between the user profile and candidate suggestions, these noisy terms could mistakenly boost the ranking of the candidate suggestion. This effect has been shown in Table 6.6. We use KL-Divergence to measure the difference between the user profile from the candidate representation. It is clear that NR is able to capture difference between the user profile and the candidate suggestion and rank the suggestion at the seventh place. On the other hand, the other representations are more similar to the candidate suggestion and incorrectly rank it at a higher place.

6.6.3 Experiments on Summary Generation

We conduct two sets of experiments to evaluate the proposed structured summary generation method.

We first evaluate the quality of the summaries generated by the proposed method. The baseline method is the snippet generation method developed by Yelp, and this method was used in one of the top ranked TREC runs [26]. To compare the results of the two methods, we develop an annotation interface as shown in Figure 6.6. There are 2,109 unique suggestions from the TREC 2013 and 2014 contextual suggesion tracks, and we generate the summary for each of them using the two methods. For each suggestion, the annotation system would present the summary generated by the two

methods, and annotators are expected to read the results and decide which one is better or choose "Hard or Impossible to Decide". Two annotators are hired for this task, and they are assigned to judge 1,300 and 1,209 suggestions respectively. There are suggestions judged by both assessors so that we can see whether judgements between the two assessors are consistent.

The comparison results are shown in Table 6.7. Among the overlapped suggestions, both annotators think that our method performs better than the baseline method for over 70% of the suggestions. Similar observations can be made for the non-overlapped suggestion set as well. Thus, it is clear that our structured summary generation method is more effective than the state of the art baseline method.

Since each structured summary contains multiple components, we also conduct experiments to evaluate the effectiveness for each component. Note that the last component is personalized and it is trivial to evaluate its effectiveness, so we focus on evaluating the first three components, i.e, opening, official introduction and review. We recruit three annotators (two of whom are the same ones as in the previous task) to assess the quality of the structured summaries. Following the same judgement strategy used by TREC, there are 5 rating levels, and 0,1,2 are mapped to non-relevant and 3,4 are mapped to relevant. The interface of the annotation system is shown in Figure 6.7. Again, there are 2,109 suggestions, and we split the job among three annotators. There are 200 suggestions assessed by all the three assessors to measure the agreement. The results are evaluated with accuracy, i.e,. the number of relevant summaries divided by the number of summaries.

Table 6.8 shows the accuracy of each section for the overlapped suggestions. It is clear that all sections have high accuracy. Among them, it seems that the official introduction are less relevant than the other two components. We also measure the agreement among the three assessors, and the agreement is around 0.5 for the official introduction, 0.7 for the opening, and 0.6 for the review component. Furthermore, Table 6.9 shows the accuracy of each component for all suggestions including the ones shared among annotators. If a suggestion is from the overlapped set, i.e., having

more than one annotation, the relevance status of a component is determined by the majority vote. Since all the suggestions are from the pool of either TREC 2013 CS track or TREC 2014 CS track, we report the accuracy for each collection separately. The observation here is similar to what we observed in the overlapped set. It is clear that both opening and review components are useful and more relevant.

6.7 Summary and Future Work

In this chapter, we have described our Android application which can track the user's context information – the location and the time. After that, we introduced our efforts on the contextual suggestion where the places and events are provided to the user based on user's preference history and the current context. We have tried two user profiling methods: using categories/ descriptions and using opinions from self and other users. The results confirm the utility of both approaches.

In the future, we think the online learning is the trend. Now our system needs to collect much opinions as the starting point. It is better to feed the algorithm once the review is available. We also would like to compare our method with general collaborative filtering. Because of the setup of the problem, we believe our method is a complementary part to CF approach where ample reviews are available.

	Category	Description (web site)	Review	Preference
Example Suggestion 1	Museum	The A Museum is the oldest Holocaust museum in the United States...	A small and **clean** museum that will take you less than an hour to see everything...	✓
Example Suggestion 2	Hotel	The B Hotel is just moments from all tourists attractions and exciting things to do in Los Angeles both for business and pleasure.. .	**Dirty** hotel, the room itself was filthy...	✗
Example Suggestion 3	Restaurant	The ambiance at C is palpable. Inside our old roadhouse, you feel like you are back in the old west with our long, long "did I say" long bar….rustic décor and welcoming taff. Makes you feel right at home the minute you walk in the door… warm and friendly like!	"Good food, **clean** restaurant" - My daughter and I enjoyed the corn dog... Women's bathroom was very **clean**, much appreciated.	✓
Example Suggestion 4	Food	Country-style comfort food including all-day breakfasts & hearty lunches served in a homey space.	Awful in every conceivable way. Bad service, **dirty** environment, and tasteless slop. 2 stars for a sort of decent beer selection.	✗
Candidate Suggestion	Hotel	Hotel Z features an outdoor pool for hotel guests only and indoor/outdoor private event space...	Great hotel! **clean** and modern...	?

Figure 6.2: An example scenario when we know the user's preferences for some suggestions and want to predict the preference for the unknown one

Original Review:

Funky little spot with a *laid-back vibe* and good chow. The chile sauce had plenty of flavor and kick, and everything seemed fresh. Service was friendly and reasonably quick, and the prices were reasonable. A bit expensive but *great food* and a great ambiance.I had the club sandwich with green chile and it was delicious. Very, very good. Party of 6 - huevos rancheros, veggie burrito, steak tacos, Mac and cheese with *green chile*. Topped off by *key lime pie*. All servings enjoyed by all. If you want ambience skip. If you want a quick, good, no frills meal, this place is for you. The *best Mexican food* Ive had in a long time. The Blue Corn Enchiladas with Green Chilis were fantastic.

FR:

The same as the the raw opinion sentences above except with removal of stop words.

SR:

chile want vibe veggie time tacos steak spot skip servings seemed sauce sandwich reasonably reasonable rancheros prices plenty place pie off no meal long huevos...

NR:

chow sauce plenty flavor kick everything Service prices bit food ambiance club sandwich chile Party burrito steak tacos Mac cheese chile lime pie servings...

RS:

best santa fe; green chile; key lime pie; great food; back vibe.

Figure 6.3: An example results of different opinion-based representations

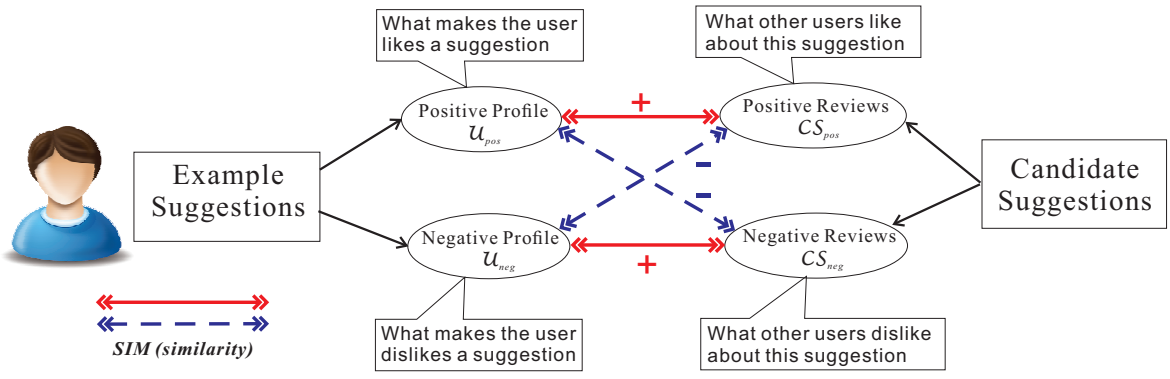


Figure 6.4: The linear interpolation method

Table 6.3: 5-fold cross validation results using linear interpolation method. $*$ (or †) indicates the improvement over the category-based (or description-based) method is statistically significant.

Collections	Methods	ERR@20	P@5
CS2012	category	0.79	0.65
	description	0.70	0.51
	FR	$0.80^{*\dagger}$	$0.68^{*\dagger}$
	SR	$0.80^{*\dagger}$	$0.66^{*\dagger}$
	NR	$0.81^{*\dagger}$	$0.66^{*\dagger}$
	RS	$0.81^{*\dagger}$	$0.67^{*\dagger}$
CS2013	category	0.66	0.68
	description	0.65	0.65
	FR	$0.72^{*\dagger}$	$0.70^{*\dagger}$
	SR	$0.71^{*\dagger}$	$0.69^{*\dagger}$
	NR	$0.71^{*\dagger}$	$0.70^{*\dagger}$
	RS	$0.69^{*\dagger}$	$0.68^{*\dagger}$
CS2014	category	0.72	0.74
	description	0.71	0.74
	FR	$0.73^{*\dagger}$	$0.76^{*\dagger}$
	SR	0.71	$0.77^{*\dagger}$
	NR	$0.75^{*\dagger}$	$0.78^{*\dagger}$
	RS	$0.75^{*\dagger}$	$0.75^{*\dagger}$
Yelp	category	0.70	0.73
	description	-	-
	FR	0.81^{*}	0.90^{*}
	SR	0.81^{*}	0.90^{*}
	NR	0.81^{*}	0.91^{*}
	RS	0.81^{*}	0.90^{*}

Table 6.4: Performance of learning to rank methods. $*$ (or †) indicates the improvement over the category-based (or description-based) method is statistically significant.

Collection	Feature	ERR@20			P@5		
		LR	LMART	MART	LR	LMART	MART
CS2012	category	0.76	0.72	0.76	0.65	0.56	0.66
	description	0.68	0.64	0.66	0.48	0.55	0.56
	FR	0.66	$0.73^{*\dagger}$	$0.80^{*\dagger}$	$0.52^{\dagger}$	$0.63^{*\dagger}$	$0.64^{\dagger}$
	SR	0.64	$0.75^{*\dagger}$	$0.73^{\dagger}$	0.47	$0.63^{*\dagger}$	0.56
	NR	0.64	$0.74^{*\dagger}$	$0.75^{\dagger}$	0.47	$0.63^{*\dagger}$	$0.61^{\dagger}$
	RS	0.61	$0.80^{*\dagger}$	$0.76^{*\dagger}$	0.45	$0.67^{*\dagger}$	$0.63^{\dagger}$
CS2013	category	0.65	0.68	0.66	0.70	0.67	0.68
	description	0.65	0.66	0.65	0.62	0.62	0.65
	FR	$0.65^{\dagger}$	$0.69^{*\dagger}$	$0.72^{*\dagger}$	$0.63^{\dagger}$	$0.68^{*\dagger}$	$0.70^{*\dagger}$
	SR	$0.65^{\dagger}$	$0.69^{*\dagger}$	$0.71^{*\dagger}$	0.57	$0.68^{*\dagger}$	$0.69^{*\dagger}$
	NR	$0.65^{\dagger}$	$0.70^{*\dagger}$	$0.71^{*\dagger}$	$0.64^{\dagger}$	$0.68^{*\dagger}$	$0.70^{*\dagger}$
	RS	$0.65^{\dagger}$	$0.69^{*\dagger}$	$0.71^{*\dagger}$	0.59	$0.70^{*\dagger}$	$0.70^{*\dagger}$
CS2014	category	0.70	0.69	0.71	0.75	0.70	0.75
	description	0.71	0.68	0.71	0.74	0.70	0.75
	FR	0.67	$0.75^{*\dagger}$	$0.76^{*\dagger}$	0.66	$0.76^{*\dagger}$	$0.79^{*\dagger}$
	SR	0.62	$0.70^{*\dagger}$	$0.75^{*\dagger}$	0.60	$0.72^{*\dagger}$	$0.78^{*\dagger}$
	NR	0.67	$0.73^{*\dagger}$	$0.75^{*\dagger}$	0.68	$0.77^{*\dagger}$	$0.79^{*\dagger}$
	RS	0.66	$0.73^{*\dagger}$	$0.74^{*\dagger}$	0.63	$0.76^{*\dagger}$	$0.79^{*\dagger}$
Yelp	category	0.69	0.67	0.68	0.72	0.56	0.72
	FR	0.78^{*}	0.77^{*}	0.78^{*}	0.84^{*}	0.76^{*}	0.89^{*}
	SR	0.77^{*}	0.80^{*}	0.79^{*}	0.85^{*}	0.81^{*}	0.93^{*}
	NR	0.80^{*}	0.76^{*}	0.80^{*}	0.85^{*}	0.77^{*}	0.93^{*}
	RS	0.79^{*}	0.76^{*}	0.79^{*}	0.85^{*}	0.73^{*}	0.92^{*}

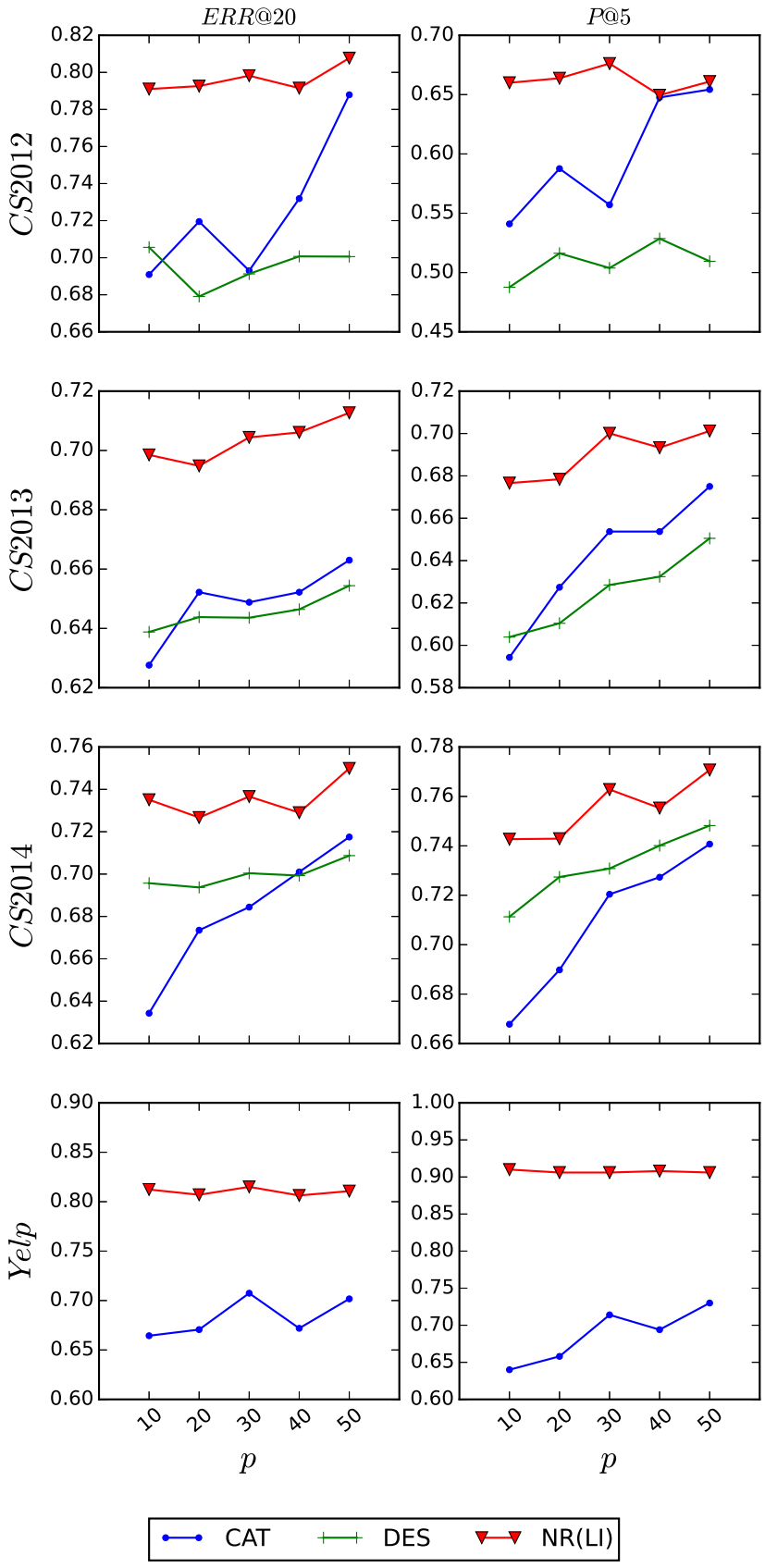


Figure 6.5: The performance of using less data to build user profile

Table 6.5: Top frequent terms in different user profiles (id:918) and positive candidate profile (id:107)

Positive User Profiles	
NR	place,burg,time,beer,food,chicago,wing,pie,art,chicken,kuma, view,bar,wait,day,drink,people,friend,table,hour,thing,cheese, sauce,night,fry
FR	burg,place,go,good,get,wait,time,great,beer,like,just,one,food, love,chicago,really,best,kuma,order,friend,will,also, back,bar,wing
SR	order,go,burg,beer,worth,wing,will,went,well,way,want,wait, visit,view,two,try,time,though,think,take,table,sure,still, something,service
RS	great,good,place,best,burg,amaze,time,favorite,beer,pie, chicago,food,art,view,first,nice,ever,delicious,beautiful,fan, awesome,worth,wait,friend,free
Positive Candidate Profile	
Name	Little Goat
Description	Upscale diner next to the Little Goat Bakery serving breakfast items, sandwiches, burgers & more
goat,wait,little,good,food,great,order,place,like,dine,time,go,menu,love,just, try,back,friend,get,really,delicious,also,one,breakfast,sandwich,cheese,got, table,pork,service,will,pancake,come,serve,coffee,well,can,amaze,definite,bread	

Table 6.6: KL divergence between positive user profile (id:918) and positive candidate profile (id:107)

Representations	KL Div.	Ranking
NR	1.54	7
FR	0.61	2
SR	1.40	2
RS	0.95	5

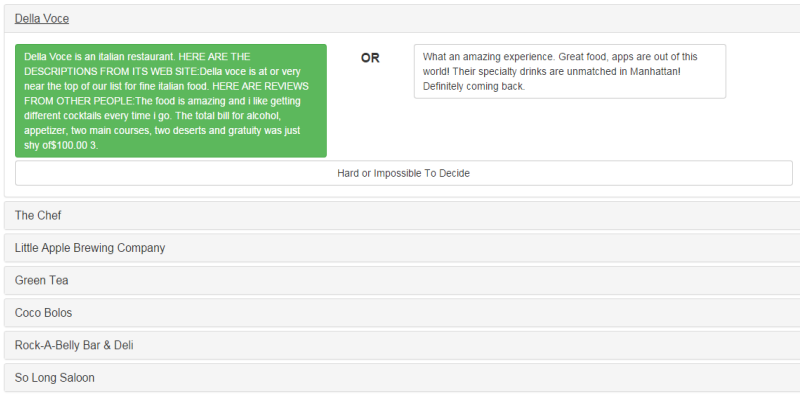


Figure 6.6: Screen shot of the web-based annotation system to compare two summary generation methods

Table 6.7: Comparison of results summarization methods

Overlapped Suggestions		
	Annotator#1	Annotator#2
Our method is better than the baseline.	71%	86%
Our method is worse than the baseline.	20%	11%
Hard or Impossible to decide	9%	4%
Non-Overlapped Suggestions		
	Annotator#1	Annotator#2
Our method is better than the baseline.	78%	68%
Our method is worse than the baseline.	15%	32%
Hard or Impossible to decide	7%	0%

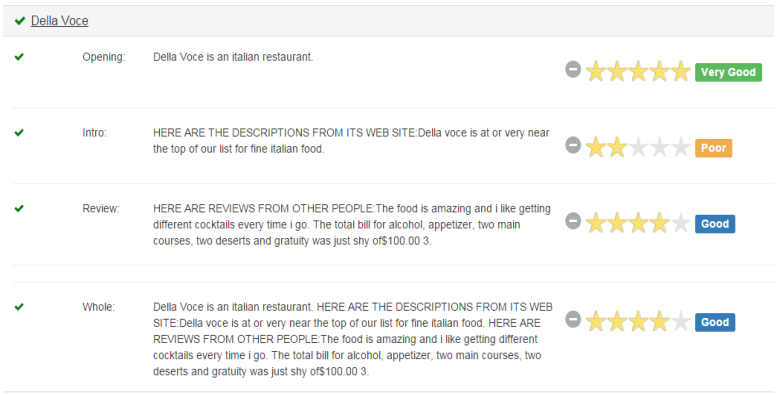


Figure 6.7: Screen shot of the web-based annotation system to evaluate the effectiveness of components

Table 6.8: Evaluation results on the overlapped suggestions (measured by accuracy)

Components	Annotator #1	Annotator #2	Annotator #3
Opening	0.98	0.81	0.80
"Official" Intro	0.75	0.53	0.78
Review	0.87	0.95	0.99

Table 6.9: Evaluation results on all the suggestions (measured by accuracy)

Components	CS2013	CS2014
Opening	0.99	0.83
"Official" Intro	0.56	0.47
Review	0.69	0.77

Chapter 7

CONCLUSION AND FUTURE WORK

In this thesis, we have introduced several IR toolkits which are designed to address the new challenges faced in front of both IR researchers and the crowds.

7.1 Conclusion

VIRLab and Anserini are mainly for IR teaching. Specifically, VIRLab removes the burden of setting up complex experimental environment to just test a simple ranking model. This greatly promotes the dissemination of IR knowledge with best practice. Anserini is built on top of Lucene. Lucene is efficient and scalable without compromising effectiveness. Furthermore, Lucene has the benefit of a large user community and broad adoption in industry. Anserini fills in the "rough edges" of using Lucene for information retrieval research by providing wrappers and extensions that simplify common tasks such as indexing large research web collections and performing standard *ad hoc* retrieval runs.

We further demonstrated the utility of reproducibility evaluation system including PPE and RISE. PPE is a prototype of more general idea of how a unified evaluation system should be designed, considering various scenarios. RISE is a more specific platform where different ranking models can be easily implemented, getting evaluated and compared. As demonstrated in the paper, we have implemented 21 retrieval functions and evaluate them over 16 TREC data sets. All the implementations and the evaluation results are available at the *RISE* platform[1].

As a pure research tool, the performance analysis of ranking models on keyword queries is proposed in this thesis. The toolkit can provide the practical performance

[1] http://rires.info:8080/

upper bound of the classical ranking models such as BM25 and Dirichlet language model for single-term queries based on cost/gain analysis whose idea is borrowed from learning to rank. The toolkit can also identify best subqueries for any keyword queries other than single-term queries. This is realized by exploring the post-retrieval features such as term proximity, term score and term scores tensors properties.

We last effort in this thesis is to provide a tool to enable the "zero query" mobile search. The aim of this tool is to return satisfying results based on not only user preference history but also contextual information. For this study, we first build a Android mobile application to track user's geographical and temporal information. Based on the user's current context, we recommend the interesting places and events to the user based on user's preference history. We propose two methods to model user profile. Category and description are firstly used and opinions are also tested. Experimental results on TREC collections confirm the utility of both approaches.

7.2 Future Work

Moving forward, we anticipate substantial continued interest at the intersection of deep learning and information retrieval, and the multi-stage ranking architecture of Anserini provides a natural integration point for future explorations.

We plan to make the unified reproducibility evaluation system publicly available as a new IR evaluation platform. All IR researchers are welcome to use the system to evaluate their models. We also plan to extend the functionality of this system to support more IR tasks.

For the tool for analyzing the ranking models, since these features for identifying the best subqueries are post-retrieval, it would inevitably increase the processing time and make it less practical to apply this method online. We plan to study more efficient ways of computing features in our future work.

For contextual suggestion, there are many interesting directions that can be pursued in the future. First, it would be interesting to evaluate the proposed method in the personalized local search problem. Second, we only focus on the user modeling,

and plan to study how to incorporate other context-related factors such as distances and recency into the ranking process.

BIBLIOGRAPHY

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

[2] Jaime Arguello, Matt Crane, Fernando Diaz, Jimmy Lin, and Andrew Trotman. Report on the SIGIR 2015 workshop on Reproducibility, Inexplicability, and Generalizability of Results (RIGOR). *SIGIR Forum*, 49(2):107–116, 2015.

[3] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. Improvements that don't add up: Ad-hoc retrieval results since 1998. In *CIKM*, pages 601–610, 2009.

[4] Nima Asadi and Jimmy Lin. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *SIGIR*, pages 997–1000, 2013.

[5] Niranjan Balasubramanian, Giridhar Kumaran, and Vitor R. Carvalho. Exploring reductions for long web queries. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 571–578, New York, NY, USA, 2010. ACM.

[6] Jie Bao, Yu Zheng, and Mohamed F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 199–208, New York, NY, USA, 2012. ACM.

[7] Alejandro Bellogín, G. Gebrekirstos Gebremeskel, Jiyin He, Jimmy Lin, and Alan Said. Cwi and tu delft notebook trec 2013: Contextual suggestion, federated web search, kba, and web tracks. In *Proceedings of TREC'13*, 2013.

[8] Michael Bendersky and W. Bruce Croft. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 491–498, New York, NY, USA, 2008. ACM.

[9] Christopher J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical report, Microsoft Research, 2010.

[10] Christopher J.C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, June 2010.

[11] C.J.C. Burges, R. Ragno, and Q.V. Le. Learning to rank with non-smooth cost functions. In *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, January 2007.

[12] Ben Carterette, Evangelos Kanoulas, Mark Hall, Ashraf Bah, and Paul Clough. Overview of the trec 2013 session track. In *Proceedings of TREC'13*, 2013.

[13] Ben Carterette, Evangelos Kanoulas, Mark Hall, Ashraf Bah, and Paul Clough. Overview of the trec 2014 session track. In *Proceedings of TREC'14*, 2014.

[14] Ben Carterette, Evangelos Kanoulas, Mark Hall, and Paul Clough. Overview of the TREC 2014 session track. In *TREC*, 2014.

[15] Ben Carterette, Virgil Pavlu, Hui Fang, and Evangelos Kanoulas. Million query track 2009 overview. In *Proceedings of TREC'09*, 2009.

[16] Marc-Allen Cartright, Samuel Huston, and H Field. Galago: A modular distributed processing and retrieval system. In *SIGIR 2012 Workshop on Open Source Information Retrieval*, pages 25–31, 2012.

[17] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 621–630, New York, NY, USA, 2009. ACM.

[18] Yan Chen and Yan-Qing Zhang. A query substitution-search result refinement approach for long query web searches. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, pages 245–251, Washington, DC, USA, 2009. IEEE Computer Society.

[19] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

[20] Charles L. A. Clarke, J. Shane Culpepper, and Alistair Moffat. Assessing efficiency—effectiveness tradeoffs in multi-stage retrieval systems without using relevance judgments. *IRJ*, 19(4):351–377, 2016.

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

[22] Kevyn Collins-Thompson, Craig Macdonald, Paul Bennett, Fernando Diaz, and Ellen M. Voorhees. Trec 2014 web track overview. In *Proceedings of TREC'14*, 2014.

[23] Ronan Cummins, Mounia Lalmas, Colm O'Riordan, and Joemon M. Jose. Navigating the user query space. In *Proceedings of the 18th International Conference on String Processing and Information Retrieval*, SPIRE'11, pages 380–385, Berlin, Heidelberg, 2011. Springer-Verlag.

[24] Adriel Dean-Hall, Charles Clarke, Jaap Kamps, Julia Kiseleva, and Ellen Voorhees. Overview of the trec 2015 contextual suggestion track. In *Proceedings of TREC'15*, 2015.

[25] Adriel Dean-Hall, Charles Clarke, Jaap Kamps, Paul Thomas, Nicole Simone, and Ellen Voorhees. Overview of the trec 2013 contextual suggestion track. In *Proceedings of TREC'13*, 2013.

[26] Adriel Dean-Hall, Charles Clarke, Jaap Kamps, Paul Thomas, Nicole Simone, and Ellen Voorhees. Overview of the trec 2014 contextual suggestion track. In *Proceedings of TREC'14*, 2014.

[27] Adriel Dean-Hall, Charles Clarke, Jaap Kamps, Paul Thomas, and Ellen Voorhees. Overview of the trec 2012 contextual suggestion track. In *Proceedings of TREC'12*, 2012.

[28] Pinar Donmez, Krysta M. Svore, and Christoper J.C. Burges. On the local optimality of lambdarank. In *SIGIR*. Association for Computing Machinery, Inc., July 2009.

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

[30] Hui Fang, Tao Tao, and Chengxiang Zhai. Diagnostic evaluation of information retrieval models. *ACM Trans. Inf. Syst.*, 29(2):7:1–7:42, April 2011.

[31] Hui Fang and Hao Wu. An exploration of query term deletion. 2011.

[32] Hui Fang, Hao Wu, Peilin Yang, and ChengXiang Zhai. Virlab: A web-based virtual lab for learning and studying information retrieval models. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 1249–1250, New York, NY, USA, 2014. ACM.

[33] Hui Fang and ChengXiang Zhai. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 480–487, New York, NY, USA, 2005. ACM.

[34] Hui Fang and ChengXiang Zhai. Virlab: A platform for privacy-preserving evaluation for information retrieval models. In *Proceeding of the 1st International Workshop on Privacy-Preserving IR:*, 2014.

[35] Nicola Ferro, Norbert Fuhr, Kalervo Järvelin, Noriko Kando, Matthias Lippold, and Justin Zobel. Increasing reproducibility in ir: Findings from the Dagstuhl seminar on "reproducibility of data-oriented experiments in e-science". *SIGIR Forum*, 50(1):68–82, 2016.

[36] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[37] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 340–348, 2010.

[38] Tim Gollub, Benno Stein, and Steven Burrows. Ousting ivory tower research: Towards a web framework for providing experiments as a service. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 1125–1126, New York, NY, USA, 2012. ACM.

[39] Allan Hanbury and Henning Müller. Automated component-level evaluation: Present and future. In *Proceedings of the 2010 International Conference on Multilingual and Multimodal Information Access Evaluation: Cross-language Evaluation Forum*, CLEF'10, pages 124–135, Berlin, Heidelberg, 2010. Springer-Verlag.

[40] N. Hariri, B. Mobasher, R. Burke, and Y. Zheng. Context-aware recommendation based on review mining. In *Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems*, 2011.

[41] Donna Harman. Overview of the third text retrieval conference (trec-3). In *Proceedings of TREC'94*, 1994.

[42] Ben He and Iadh Ounis. Inferring query performance using pre-retrieval predictors. In *In Proc. Symposium on String Processing and Information Retrieval*, pages 43–54. Springer Verlag, 2004.

[43] Ben He and Iadh Ounis. A study of the dirichlet priors for term frequency normalisation. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 465–471, New York, NY, USA, 2005. ACM.

[44] Frank Hopfgartner, Allan Hanbury, Henning Müller, Noriko Kando, Simon Mercer, Jayashree Kalpathy-Cramer, Martin Potthast, Tim Gollub, Anastasia

Krithara, Jimmy Lin, Krisztian Balog, and Ivan Eggel. Report on the evaluation-as-a-service (eaas) expert workshop. *SIGIR Forum*, 49(1):57–65, June 2015.

[45] Xiangdong Huang, Ziyang Yan, Senxue Jing, Hongwei Fang, and Li Xiao. Co-prime sensing-based frequency estimation using reduced single-tone snapshots. *Circuits, Systems, and Signal Processing*, 35(9):3355–3366, 2016.

[46] Gilles Hubert and Guillaume Cabanac. Irit at trec 2012 contextual suggestion track. In *Proceedings of TREC'12*, 2012.

[47] Samuel Huston and W. Bruce Croft. Evaluating verbose query processing techniques. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 291–298, New York, NY, USA, 2010. ACM.

[48] Niklas Jakob, Stefan Hagen Weber, Mark Christoph Müller, and Iryna Gurevych. Beyond the stars: Exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceedings of the 1st International CIKM Workshop on Topic-sentiment Analysis for Mass Opinion*, TSA '09, pages 57–64, New York, NY, USA, 2009. ACM.

[49] Marijn Koolen, Hugo Huurdeman, and Jaap Kamps. University of amsterdam at the trec 2013 contextual suggestion track: Learning user preferences from wikitravel categories. In *Proceedings of TREC'13*, 2013.

[50] Giridhar Kumaran and James Allan. A case for shorter queries, and helping users create them. In *HLT-NAACL*, pages 220–227, 2007.

[51] Giridhar Kumaran and James Allan. Effective and efficient user interaction for long queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 11–18, New York, NY, USA, 2008. ACM.

[52] Giridhar Kumaran and Vitor R. Carvalho. Reducing long queries using query quality predictors. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 564–571, New York, NY, USA, 2009. ACM.

[53] Dmitry Lagun and Eugene Agichtein. Viewser: Enabling large-scale remote user studies of web search examination and interaction. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 365–374, New York, NY, USA, 2011. ACM.

[54] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *SIGIR*, pages 120–127, 2001.

[55] Asher Levi, Osnat Mokryn, Christophe Diot, and Nina Taft. Finding a needle in a haystack of reviews: cold start context-based hotel recommender system. In *Proceedings of the RecSys'12*, 2012.

[56] Hanchen Li, Zhen Yang, Yingxu Lai, Lijuan Duan, and Kefeng Fan. Bjut at trec 2014 contextual suggestion track: Hybrid recommendation based on open-web information. In *Proceedings of TREC'14*, 2014.

[57] Hua Li and Rafael Alonso. User modeling for contextual suggestion. In *Proceedings of TREC'14*, 2014.

[58] Jimmy Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig MacDonald, and Sebastiano Vigna. Toward reproducible baselines: The open-source ir reproducibility challenge. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *ECIR*, volume 9626 of *Lecture Notes in Computer Science*, pages 408–420. Springer, 2016.

[59] Jimmy Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig Macdonald, and Sebastiano Vigna. Toward reproducible baselines: The open-source ir reproducibility challenge. In *European Conference on Information Retrieval*, pages 408–420. Springer, 2016.

[60] Jimmy Lin and Miles Efron. Evaluation as a service for information retrieval. *SIGIR Forum*, 47(2):8–14, January 2013.

[61] Jimmy Lin and Miles Efron. Infrastructure support for evaluation as a service. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 79–82, New York, NY, USA, 2014. ACM.

[62] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. Overview of the trec-2014 microblog track. In *Proceedings of TREC'14*, 2014.

[63] Jimmy Lin, Miles Efron, Yulu Wang, Garrick Sherman, and Ellen Voorhees. Overview of the trec-2015 microblog track. In *Proceedings of TREC'15*, 2015.

[64] Jimmy Lin, Donald Metzler, Tamer Elsayed, and Lidan Wang. Of Ivory and Smurfs: Loxodontan MapReduce experiments for web search. In *TREC*, 2009.

[65] Tie-Yan Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, March 2009.

[66] Xitong Liu and Hui Fang. Latent entity space: a novel retrieval approach for entity-bearing queries. *Information Retrieval Journal*, 18(6):473–503, 2015.

[67] Xitong Liu, Peilin Yang, and Hui Fang. Entexpo: An interactive search system for entity-bearing queries. In *European Conference on Information Retrieval*, pages 784–788. Springer International Publishing, 2014.

[68] Xitong Liu, Peilin Yang, and Hui Fang. Entity came to rescue-leveraging entities to minimize risks in web search. Technical report, DTIC Document, 2014.

[69] Yuanhua Lv and ChengXiang Zhai. Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 7–16, New York, NY, USA, 2011. ACM.

[70] Craig Macdonald, Rodrygo L. Santos, and Iadh Ounis. The whens and hows of learning to rank for web search. *Inf. Retr.*, 16(5):584–628, October 2013.

[71] Richard McCreadie, Romain Deveaud, M-Dyaa Albakour, Stuart Mackie, Nut Limsopatham, Craig Macdonald, Iadh Ounis, Thibaut Thonet, and Bekir Taner. University of glasgow at trec 2014: Experiments with terrier in contextual suggestion, temporal summarisation and web tracks. In *Proceedings of TREC'14*, 2014.

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

[73] Donald Metzler, Victor Lavrenko, and W. Bruce Croft. Formal multiple-bernoulli models for language modeling. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 540–541, New York, NY, USA, 2004. ACM.

[74] Hannes Mühleisen, Thaer Samar, Jimmy Lin, and Arjen de Vries. Old dogs are great at new tricks: Column stores for ir prototyping. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 863–866, New York, NY, USA, 2014. ACM.

[75] Hannes Mühleisen, Thaer Samar, Jimmy Lin, and Arjen de Vries. Old dogs are great at new tricks: Column stores for ir prototyping. In *SIGIR*, pages 863–866, 2014.

[76] Anastasios Noulas, Salvatore Scellato, Neal Lathia, and Cecilia Mascolo. A random walk around the city: New venue recommendation in location-based social networks. In *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust*, SOCIALCOM-PASSAT '12, pages 144–153, Washington, DC, USA, 2012. IEEE Computer Society.

[77] Jiaul H. Paik. A novel tf-idf weighting scheme for effective ranking. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 343–352, New York, NY, USA, 2013. ACM.

[78] Jiaul H. Paik and Jimmy Lin. Retrievability in api-based "evaluation as a service". In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 91–94, New York, NY, USA, 2016. ACM.

[79] Jae Hyun Park and W. Bruce Croft. Query term ranking based on dependency parsing of verbose queries. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 829–830, New York, NY, USA, 2010. ACM.

[80] Jae Hyun Park, W. Bruce Croft, and David A. Smith. A quasi-synchronous dependence model for information retrieval. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 17–26, New York, NY, USA, 2011. ACM.

[81] Jan Pedersen. Query understanding at bing. In *Invited Talk at SIGIR*, 2010.

[82] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 275–281, New York, NY, USA, 1998. ACM.

[83] Rani Qumsiyeh and Yiu-Kai Ng. Predicting the ratings of multimedia items for making personalized recommendations. In *Proceedings of SIGIR'12*, 2012.

[84] Sindhu Raghavan, Suriya Gunasekar, and Joydeep Ghosh. Review quality aware collaborative filtering. In *Proceedings of RecSys'12*, 2012.

[85] Ashwani Rao and Ben Carterette. Udel at trec 2012. In *Proceedings of TREC'12*, 2012.

[86] Jinfeng Rao, Jimmy Lin, and Miles Efron. Reproducible experiments on lexical and temporal feedback for tweet search. In Allan Hanbury, Gabriella Kazai, Andreas Rauber, and Norbert Fuhr, editors, *Advances in Information Retrieval*, volume 9022 of *Lecture Notes in Computer Science*, pages 755–767. Springer International Publishing, 2015.

[87] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. pages 109–126, 1996.

[88] Dwaipayan Roy, Ayan Bandyopadhyay, and Mandar Mitra. A simple context dependent suggestion system. In *Proceedings of TREC'13*, 2013.

[89] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[90] Anna Shtok, Oren Kurland, David Carmel, Fiana Raiber, and Gad Markovits. Predicting query performance by query-drift estimation. *ACM Trans. Inf. Syst.*, 30(2):11:1–11:35, May 2012.

[91] Luo Si and Hui Yang. Privacy-preserving ir: when information retrieval meets privacy and security. In *Proceedings of the SIGIR'14*, 2014.

[92] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 21–29, New York, NY, USA, 1996. ACM.

[93] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009.

[94] Tao Tao and ChengXiang Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. SIGIR '06, pages 162–169, New York, NY, USA, 2006. ACM.

[95] Michael Taylor, Hugo Zaragoza, Nick Craswell, Stephen Robertson, and Chris Burges. Optimisation methods for ranking functions with multiple parameters. CIKM '06, pages 585–593, New York, NY, USA, 2006. ACM.

[96] Andrew Trotman, Antti Puurula, and Blake Burgess. Improvements to bm25 and language models examined. In *Proceedings of the 19th Australasian Document Computing Symposium*, ADCS '14, Melbourne, VIC, Australia, 2014. ACM.

[97] Andrew Trotman, Antti Puurula, and Blake Burgess. Improvements to BM25 and language models examined. In *ADCS*, pages 58–65, 2014.

[98] Ellen M. Voorhees. Overview of the trec 2004 robust retrieval track. In *Proceedings of TREC'04*, 2004.

[99] Ellen M. Voorhees. Overview of the trec 2005 robust retrieval track. In *Proceedings of TREC'05*, 2005.

[100] Ellen M. Voorhees, Shahzad Rajput, and Ian Soboroff. Promoting repeatability through open runs. In *EVIA*, pages 17–20, 2016.

[101] Lidan Wang, Jimmy Lin, and Donald Metzler. Learning to efficiently rank. In *Proceedings of SIGIR'10*.

[102] Lidan Wang, Jimmy Lin, and Donald Metzler. A cascade ranking model for efficient ranked retrieval. In *SIGIR*, pages 105–114, 2011.

[103] Xiaobing Xue, Samuel Huston, and W. Bruce Croft. Improving verbose queries using subset distribution. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1059–1068, New York, NY, USA, 2010. ACM.

[104] Bishan Yang, Nish Parikh, Gyanit Singh, and Neel Sundaresan. A study of query term deletion using large-scale e-commerce search logs. In *Proceedings of the 36th European Conference on IR Research on Advances in Information Retrieval - Volume 8416*, ECIR 2014, pages 235–246, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[105] Peilin Yang and Hui Fang. An exploration of ranking-based strategy for contextual suggestion. In *Proceedings of TREC'12*, 2012.

[106] Peilin Yang and Hui Fang. An opinion-aware approach to contextual suggestion. In *Proceedings of TREC'13*, 2013.

[107] Peilin Yang and Hui Fang. Opinion-based user profile modeling for contextual suggestions. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval*, ICTIR '13, pages 18:80–18:83, New York, NY, USA, 2013. ACM.

[108] Peilin Yang and Hui Fang. Exploration of opinion-aware approach to contextual suggestion. In *Proceedings of TREC'14*, 2014.

[109] Peilin Yang and Hui Fang. Combining opinion profile modeling with complex contextfiltering for contextual suggestion. In *Proceedings of TREC'15*, 2015.

[110] Peilin Yang and Hui Fang. Estimating retrieval performance bound for single term queries. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 237–240, New York, NY, USA, 2016. ACM.

[111] Peilin Yang and Hui Fang. A reproducibility study of information retrieval models. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 77–86, New York, NY, USA, 2016. ACM.

[112] Peilin Yang and Hui Fang. Towards privacy-preserving evaluation for information retrieval models over industry data sets. In *Proceedings of the 2017 ACM International Conference on the Theory of Information Retrieval*, ICTIR '17, New York, NY, USA, 2017. ACM.

[113] Peilin Yang, Hui Fang, and Jimmy Lin. Reinvigorating the use of lucene for information retrieval research. In *Proceedings of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, New York, NY, USA, 2017. ACM.

[114] Peilin Yang, Hongning Wang, Hui Fang, and Deng Cai. Opinions matter: a general approach to user profile modeling for contextual suggestion. *Information Retrieval Journal*, 18(6):586–610, 2015.

[115] Andrew Yates, Dave DeBoer, Hui Yang, Nazli Goharian, Steve Kunath, and Ophir Frieder. (not too) personalized learning to rank for contextual suggestion. In *Proceedings of TREC'12*, 2012.

[116] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. SIGIR '07, pages 271–278, New York, NY, USA, 2007. ACM.

[117] ChengXiang Zhai and John Lafferty. Two-stage language models for information retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 49–56, New York, NY, USA, 2002. ACM.

[118] 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, April 2004.

Appendix

COPYRIGHTS

Please see the following pages for the copyrights of the related contents in this thesis.

ACM Copyright and Audio/Video Release

Title of the Work: Opinion-based User Profile Modeling for Contextual Suggestions

Publication and/or Conference Name: Proceedings of the 2013 International Conference on the Theory of Information Retrieval Proceedings

Author/Presenter(s): Peilin Yang;Hui Fang

Auxiliary Materials (provide filenames and a description of auxiliary content, if any, for display in the ACM Digital Library. The description may be provided as a ReadMe file):

I. Copyright Transfer, Reserved Rights and Permitted Uses

* Your Copyright Transfer is conditional upon you agreeing to the terms set out below.

Copyright to the Work and to any supplemental files integral to the Work which are submitted with it for review and publication such as an extended proof, a PowerPoint outline, or appendices that may exceed a printed page limit, (including without limitation, the right to publish the Work in whole or in part in any and all forms of media, now or hereafter known) is hereby transferred to the ACM (for Government work, to the extent transferable) effective as of the date of this agreement, on the understanding that the Work has been accepted for publication by ACM.

Reserved Rights and Permitted Uses

(a) All rights and permissions the author has not granted to ACM are reserved to the Owner, including all other proprietary rights such as patent or trademark rights.

(b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM, Owner shall have the right to do the following:

(i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media.

(ii) Create a "Major Revision" which is wholly owned by the author

(iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, or (3) any repository legally mandated by an agency funding the research on which the Work is based.

(iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository;

(v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers;

(vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable;

(vii) Make free distributions of the published Version of Record for Classroom and

Personal Use;

(viii) Bundle the Work in any of Owner's software distributions; and

(ix) Use any Auxiliary Material independent from the Work.

Authors should understand that consistent with ACMs policy of encouraging dissemination of information, each work published by ACM appears with the ACM copyright and the following notice:

"Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org."

☑ A. Assent to Assignment. I hereby represent and warrant that I am the sole owner (or authorized agent of the copyright owner(s)), with the exception of third party materials detailed in section III below. I have obtained permission for any third-party material included in the Work.

☐ B. Declaration for Government Work. I am an employee of the National Government of my country and my Government claims rights to this work, or it is not copyrightable (Government work is classified as Public Domain in U.S. only)

Are any of the co-authors, employees or contractors of a National Government?
○ Yes ◉ No
Country:

II. PERMISSION FOR CONFERENCE TAPING AND DISTRIBUTION (Check A and, if applicable, B)
A. Audio /Video Release

I hereby grant permission for ACM to include my name, likeness, presentation and comments in any and all forms, for the Conference and/or Publication.

I further grant permission for ACM to record and/or transcribe and reproduce my presentation as part of the ACM Digital Library, and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known.

I understand that my presentation will not be sold separately by itself as a stand-alone product without my direct consent. Accordingly, I give ACM the right to use my image, voice, pronouncements, likeness, and my name, and any biographical material submitted by me, in connection with the Conference and/or Publication, whether used in excerpts or in full, for distribution described above and for any associated advertising or exhibition.

Do you agree to the above Audio/Video Release? (•) Yes ( ) No

B. Auxiliary Materials, not integral to the Work

I hereby grant ACM permission to serve files named below containing my Auxiliary Material from the ACM Digital Library. I hereby represent and warrant that my Auxiliary Material contains no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software, and I hereby agree to indemnify and hold harmless ACM from all liability, losses, damages, penalties, claims, actions, costs and expenses (including reasonable legal expense) arising from the use of such files.

[x] I agree to the above Auxiliary Materials permission statement

III. Third Party Materials

In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below.

(•) We/I have not used third-party material.
( ) We/I have used third-party materials and have necessary permissions.

IV. Artistic Images

If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part IV and be sure to include a notice of copyright with each such image in the paper.
(•) We/I do not have any artistic images.
( ) We/I have any artistic images.

V. Representations, Warranties and Covenants

The undersigned hereby represents, warrants and covenants as follows:

(a) Owner is the sole owner or authorized agent of Owner(s) of the Work;

(b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM;

(c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit;

(d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI

pointers on any such prior postings;

(e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and

(f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version.

☑ I agree to the Representations, Warranties and Covenants

DATE: **06/28/2013** sent to yangpeilyn@gmail.com at **08:06:21**

ACM Copyright and Audio/Video Release

Title of the Work: Retrieval Performance Bound Analysis for Single Term Queries
Submission ID:ictir168s
Author/Presenter(s): Peilin Yang (Univ. of Delaware); Hui Fang (Univ. of Delaware)
Type of material: Short Paper

Publication and/or Conference Name: ICTIR '16: ACM SIGIR International Conference on the Theory of Information Retrieval Proceedings

I. Copyright Transfer, Reserved Rights and Permitted Uses

* Your Copyright Transfer is conditional upon you agreeing to the terms set out below.

Copyright to the Work and to any supplemental files integral to the Work which are submitted with it for review and publication such as an extended proof, a PowerPoint outline, or appendices that may exceed a printed page limit, (including without limitation, the right to publish the Work in whole or in part in any and all forms of media, now or hereafter known) is hereby transferred to the ACM (for Government work, to the extent transferable) effective as of the date of this agreement, on the understanding that the Work has been accepted for publication by ACM.

Reserved Rights and Permitted Uses

(a) All rights and permissions the author has not granted to ACM are reserved to the Owner, including all other proprietary rights such as patent or trademark rights.

(b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM, Owner shall have the right to do the following:

(i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media.

(ii) Create a "Major Revision" which is wholly owned by the author

(iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency funding the research on which the Work is based, and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles.

(iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository;

(v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers;

(vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable;

(vii) Make free distributions of the published Version of Record for Classroom and Personal Use;

(viii) Bundle the Work in any of Owner's software distributions; and

(ix) Use any Auxiliary Material independent from the Work.

When preparing your paper for submission using the ACM TeX templates, the rights and permissions information and the bibliographic strip must appear on the lower left hand portion of the first page.

The new Authorized ACM TeX template .cls version 2.8, automatically creates and positions these text blocks for you based on the code snippet which is system-generated based on your rights management choice and this particular conference.

Please copy and paste the following code snippet into your TeX file between \begin{document} and \maketitle, either after or before CCS codes.

```
\CopyrightYear{2016}
\setcopyright{acmcopyright}
\conferenceinfo{ICTIR '16,}{September 12-16, 2016, Newark, DE, USA}
\isbn{978-1-4503-4497-5/16/09}\acmPrice{\$15.00}
\doi{http://dx.doi.org/10.1145/2970398.2970428}
```

If you are using the ACM Microsoft Word template, or still using an older version of the ACM TeX template, or the current versions of the ACM SIGCHI, SIGGRAPH, or SIGPLAN TeX templates, you must copy and paste the following text block into your document as per the instructions provided with the templates you are using:

ICTIR '16, September 12-16, 2016, Newark, DE, USA
© 2016 ACM. ISBN 978-1-4503-4497-5/16/09...$15.00
DOI: http://dx.doi.org/10.1145/2970398.2970428

NOTE: Make sure to include your article's DOI as part of the bibstrip data; DOIs will be registered and become active shortly after publication in the ACM Digital Library

☑ A. Assent to Assignment. I hereby represent and warrant that I am the sole owner (or authorized agent of the copyright owner(s)), with the exception of third party materials detailed in section III below. I have obtained permission for any third-party material included in the Work.

☐ B. Declaration for Government Work. I am an employee of the National Government of my country and my Government claims rights to this work, or it is not copyrightable (Government work is classified as Public Domain in U.S. only)

Country:

Are you authorized to transfer copyright? ○ Yes ◉ No

II. PERMISSION FOR CONFERENCE TAPING AND DISTRIBUTION

Audio/Video Release
* Your Audio/Video Release is conditional upon you agreeing to the terms set out below.

I hereby grant permission for ACM to include my name, likeness, presentation and comments in any and all forms, for the Conference and/or Publication.

I further grant permission for ACM to record and/or transcribe and reproduce my presentation as part of the ACM Digital Library, and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known.

I understand that my presentation will not be sold separately as a stand-alone product without my direct consent. Accordingly, I give ACM the right to use my image, voice, pronouncements, likeness, and my name, and any biographical material submitted by me, in connection with the Conference and/or Publication, whether used in excerpts or in full, for distribution described above and for any associated advertising or exhibition.

Do you agree to the above Audio/Video Release? ◉ Yes ○ No

III. Auxiliary Material

Do you have any Auxiliary Materials? ○ Yes ◉ No

IV. Third Party Materials

In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below.

◉ We/I have not used third-party material.
○ We/I have used third-party materials and have necessary permissions.

V. Artistic Images

If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part V and be sure to include a notice of copyright with each such image in the paper.
◉ We/I do not have any artistic images.
○ We/I have any artistic images.

VI. Representations, Warranties and Covenants

The undersigned hereby represents, warrants and covenants as follows:

(a) Owner is the sole owner or authorized agent of Owner(s) of the Work;

(b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM;

(c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit;

(d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings;

(e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and

(f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version.

☑ I agree to the Representations, Warranties and Covenants

DATE: **07/11/2016** sent to franklyn@udel.edu at **23:07:25**

ACM Copyright and Audio/Video Release

Title of the Work: A Reproducibility Study of Information Retrieval Models
Submission ID:ictir152
Author/Presenter(s): Peilin Yang (Univ. of Delaware); Hui Fang (Univ. of Delaware)
Type of material: Full Paper

Publication and/or Conference Name: ICTIR '16: ACM SIGIR International Conference on the Theory of Information Retrieval Proceedings

I. Copyright Transfer, Reserved Rights and Permitted Uses

* Your Copyright Transfer is conditional upon you agreeing to the terms set out below.

Copyright to the Work and to any supplemental files integral to the Work which are submitted with it for review and publication such as an extended proof, a PowerPoint outline, or appendices that may exceed a printed page limit, (including without limitation, the right to publish the Work in whole or in part in any and all forms of media, now or hereafter known) is hereby transferred to the ACM (for Government work, to the extent transferable) effective as of the date of this agreement, on the understanding that the Work has been accepted for publication by ACM.

Reserved Rights and Permitted Uses

(a) All rights and permissions the author has not granted to ACM are reserved to the Owner, including all other proprietary rights such as patent or trademark rights.

(b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM, Owner shall have the right to do the following:

(i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media.

(ii) Create a "Major Revision" which is wholly owned by the author

(iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency funding the research on which the Work is based, and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles.

(iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository;

(v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers;

(vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable;

(vii) Make free distributions of the published Version of Record for Classroom and Personal Use;

(viii) Bundle the Work in any of Owner's software distributions; and

(ix) Use any Auxiliary Material independent from the Work.

When preparing your paper for submission using the ACM TeX templates, the rights and permissions information and the bibliographic strip must appear on the lower left hand portion of the first page.

The new Authorized ACM TeX template .cls version 2.8, automatically creates and positions these text blocks for you based on the code snippet which is system-generated based on your rights management choice and this particular conference.

Please copy and paste the following code snippet into your TeX file between \begin{document} and \maketitle, either after or before CCS codes.

```
\CopyrightYear{2016}
\setcopyright{acmcopyright}
\conferenceinfo{ICTIR '16,}{September 12-16, 2016, Newark, DE, USA}
\isbn{978-1-4503-4497-5/16/09}\acmPrice{\$15.00}
\doi{http://dx.doi.org/10.1145/2970398.2970415}
```

If you are using the ACM Microsoft Word template, or still using an older version of the ACM TeX template, or the current versions of the ACM SIGCHI, SIGGRAPH, or SIGPLAN TeX templates, you must copy and paste the following text block into your document as per the instructions provided with the templates you are using:

ICTIR '16, September 12-16, 2016, Newark, DE, USA
© 2016 ACM. ISBN 978-1-4503-4497-5/16/09...$15.00
DOI: http://dx.doi.org/10.1145/2970398.2970415

NOTE: Make sure to include your article's DOI as part of the bibstrip data; DOIs will be registered and become active shortly after publication in the ACM Digital Library

☑ A. Assent to Assignment. I hereby represent and warrant that I am the sole owner (or authorized agent of the copyright owner(s)), with the exception of third party materials detailed in section III below. I have obtained permission for any third-party material included in the Work.

☐ B. Declaration for Government Work. I am an employee of the National Government of my country and my Government claims rights to this work, or it is not copyrightable (Government work is classified as Public Domain in U.S. only)

Are any of the co-authors, employees or contractors of a National Government? ○ Yes ◉ No

Country:

II. PERMISSION FOR CONFERENCE TAPING AND DISTRIBUTION

Audio/Video Release

* Your Audio/Video Release is conditional upon you agreeing to the terms set out below.

I hereby grant permission for ACM to include my name, likeness, presentation and comments in any and all forms, for the Conference and/or Publication.

I further grant permission for ACM to record and/or transcribe and reproduce my presentation as part of the ACM Digital Library, and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known.

I understand that my presentation will not be sold separately as a stand-alone product without my direct consent. Accordingly, I give ACM the right to use my image, voice, pronouncements, likeness, and my name, and any biographical material submitted by me, in connection with the Conference and/or Publication, whether used in excerpts or in full, for distribution described above and for any associated advertising or exhibition.

Do you agree to the above Audio/Video Release? (•) Yes ( ) No

III. Auxiliary Material

Do you have any Auxiliary Materials? ( ) Yes (•) No

IV. Third Party Materials

In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below.

(•) We/I have not used third-party material.
( ) We/I have used third-party materials and have necessary permissions.

V. Artistic Images

If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part V and be sure to include a notice of copyright with each such image in the paper.

(•) We/I do not have any artistic images.
( ) We/I have any artistic images.

VI. Representations, Warranties and Covenants

The undersigned hereby represents, warrants and covenants as follows:

(a) Owner is the sole owner or authorized agent of Owner(s) of the Work;

(b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM;

(c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit;

(d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings;

(e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and

(f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version.

☑ I agree to the Representations, Warranties and Covenants

DATE: **07/11/2016** sent to franklyn@udel.edu at **23:07:28**

ACM Publishing License and Audio/Video Release

Title of the Work: Reinvigorating the use of Lucene for Information Retrieval Research
Submission ID:sp265
Author/Presenter(s): Peilin Yang (Univ. of Delaware); Hui Fang (Univ. of Delaware); Jimmy Lin (Univ. of Waterloo)
Type of material: Short paper

Publication and/or Conference Name: SIGIR '17: The 40th International ACM SIGIR Conference on Research and Development in Information Retrieval CD-ROM Proceedings

1. Glossary

2. Grant of Rights

(a) Owner hereby grants to ACM an exclusive, worldwide, royalty-free, perpetual, irrevocable, transferable and sublicenseable license to publish, reproduce and distribute all or any part of the Work in any and all forms of media, now or hereafter known, including in the above publication and in the ACM Digital Library, and to authorize third parties to do the same.

(b) In connection with software and "Artistic Images and "Auxiliary Materials, Owner grants ACM non-exclusive permission to publish, reproduce and distribute in any and all forms of media, now or hereafter known, including in the above publication and in the ACM Digital Library.

(c) In connection with any "Minor Revision", that is, a derivative work containing less than twenty-five percent (25%) of new substantive material, Owner hereby grants to ACM all rights in the Minor Revision that Owner grants to ACM with respect to the Work, and all terms of this Agreement shall apply to the Minor Revision.

☑ A. Grant of Rights. I grant the rights and agree to the terms described above.

☐ B. Declaration for Government Work. I am an employee of the national government of my country and my Government claims rights to this work, or it is not copyrightable (Government work is classified as Public Domain in U.S. only)

Are any of the co-authors, employees or contractors of a National Government? ○ Yes ◉ No

---

3. Reserved Rights and Permitted Uses.

(a) All rights and permissions the author has not granted to ACM in Paragraph 2 are reserved to the Owner, including without limitation the ownership of the copyright of the Work and all other proprietary rights such as patent or trademark rights.

(b) Furthermore, notwithstanding the exclusive rights the Owner has granted to ACM in Paragraph 2(a), Owner shall have the right to do the following:

(i) Reuse any portion of the Work, without fee, in any future works written or edited by the Author, including books, lectures and presentations in any and all media.

(ii) Create a "Major Revision" which is wholly owned by the author

(iii) Post the Accepted Version of the Work on (1) the Author's home page, (2) the Owner's institutional repository, (3) any repository legally mandated by an agency

funding the research on which the Work is based, and (4) any non-commercial repository or aggregation that does not duplicate ACM tables of contents, i.e., whose patterns of links do not substantially duplicate an ACM-copyrighted volume or issue. Non-commercial repositories are here understood as repositories owned by non-profit organizations that do not charge a fee for accessing deposited articles and that do not sell advertising or otherwise profit from serving articles.

(iv) Post an "Author-Izer" link enabling free downloads of the Version of Record in the ACM Digital Library on (1) the Author's home page or (2) the Owner's institutional repository;

(v) Prior to commencement of the ACM peer review process, post the version of the Work as submitted to ACM ("Submitted Version" or any earlier versions) to non-peer reviewed servers;

(vi) Make free distributions of the final published Version of Record internally to the Owner's employees, if applicable;

(vii) Make free distributions of the published Version of Record for Classroom and Personal Use;

(viii) Bundle the Work in any of Owner's software distributions; and

(ix) Use any Auxiliary Material independent from the Work.

When preparing your paper for submission using the ACM TeX templates, the rights and permissions information and the bibliographic strip must appear on the lower left hand portion of the first page.

The new ACM Consolidated TeX template Version 1.3x automatically creates and positions these text blocks for you based on the code snippet which is system-generated based on your rights management choice and this particular conference.

Please copy and paste the following code snippet into your TeX file between \begin{document} and \maketitle, either after or before CCS codes.

```
\copyrightyear{2017}
\acmYear{2017}
\setcopyright{acmlicensed}
\acmConference{SIGIR '17}{August 07-11, 2017}{Shinjuku, Tokyo,
Japan}\acmPrice{15.00}\acmDOI{http://dx.doi.org/10.1145/3077136.3080721}
\acmISBN{978-1-4503-5022-8/17/08}
```

ACM TeX template .cls version 2.8, automatically creates and positions these text blocks for you based on the code snippet which is system-generated based on your rights management choice and this particular conference.
Please copy and paste the following code snippet into your TeX file between \begin{document} and \maketitle, either after or before CCS codes.

```
\CopyrightYear{2017}
\setcopyright{acmlicensed}
\conferenceinfo{SIGIR '17,}{August 07-11, 2017, Shinjuku, Tokyo, Japan}
```

\isbn{978-1-4503-5022-8/17/08}\acmPrice{$15.00}
\doi{http://dx.doi.org/10.1145/3077136.3080721}

If you are using the ACM Microsoft Word template, or still using an older version of the ACM TeX template, or the current versions of the ACM SIGCHI, SIGGRAPH, or SIGPLAN TeX templates, you must copy and paste the following text block into your document as per the instructions provided with the templates you are using:

SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan

ACM ISBN 978-1-4503-5022-8/17/08...$15.00
http://dx.doi.org/10.1145/3077136.3080721

NOTE: Make sure to include your article's DOI as part of the bibstrip data; DOIs will be registered and become active shortly after publication in the ACM Digital Library

---

4. ACM Citation and Digital Object Identifier.

(a) In connection with any use by the Owner of the Definitive Version, Owner shall include the ACM citation and ACM Digital Object Identifier (DOI).
(b) In connection with any use by the Owner of the Submitted Version (if accepted) or the Accepted Version or a Minor Revision, Owner shall use best efforts to display the ACM citation, along with a statement substantially similar to the following:

> "© [Owner] [Year]. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive version was published in {Source Publication}, http://dx.doi.org/10.1145/{number}."

5. Audio/Video Recording

I hereby grant permission for ACM to include my name, likeness, presentation and comments in any and all forms, for the Conference and/or Publication.

I further grant permission for ACM to record and/or transcribe and reproduce my presentation as part of the ACM Digital Library, and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known.

I understand that my presentation will not be sold separately as a stand-alone product without my direct consent. Accordingly, I give ACM the right to use my image, voice,

pronouncements, likeness, and my name, and any biographical material submitted by me, in connection with the Conference and/or Publication, whether used in excerpts or in full, for distribution described above and for any associated advertising or exhibition.

Do you agree to the above Audio/Video Release? (•) Yes ( ) No

6. Auxiliary Material

Do you have any Auxiliary Materials? ( ) Yes (•) No

7. Third Party Materials

In the event that any materials used in my presentation or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below.

(•) We/I have not used third-party material.
( ) We/I have used third-party materials and have necessary permissions.

8. Artistic Images

If your paper includes images that were created for any purpose other than this paper and to which you or your employer claim copyright, you must complete Part IV and be sure to include a notice of copyright with each such image in the paper.
(•) We/I do not have any artistic images.
( ) We/I have any artistic images.

---

9. Representations, Warranties and Covenants

The undersigned hereby represents, warrants and covenants as follows:

(a) Owner is the sole owner or authorized agent of Owner(s) of the Work;

(b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM;

(c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit;

(d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings;

(e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and

(f) The Artistic Images, if any, are clearly and accurately noted as such (including any

applicable copyright notice) in the Submitted Version.

☑ I agree to the Representations, Warranties and Covenants.

---

10. Enforcement.

At ACM's expense, ACM shall have the right (but not the obligation) to defend and enforce the rights granted to ACM hereunder, including in connection with any instances of plagiarism brought to the attention of ACM. Owner shall notify ACM in writing as promptly as practicable upon becoming aware that any third party is infringing upon the rights granted to ACM, and shall reasonably cooperate with ACM in its defense or enforcement.

---

11. Governing Law

This Agreement shall be governed by, and construed in accordance with, the laws of the state of New York applicable to contracts entered into and to be fully performed therein.

Funding Agents

1. Natural Sciences and Engineering Research Council of Canada award number(s):

2. National Science Foundation award number(s):IIS-1423002

DATE: **04/24/2017** sent to yangpeilyn@gmail.com at **19:04:46**