# EXPLOITING CONTEXT FOR LONG-TERM INFORMATION RETRIEVAL RELATED TASKS

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 Computer Engineering

Fall 2016

# EXPLOITING CONTEXT FOR LONG-TERM

# INFORMATION RETRIEVAL RELATED TASKS

by

Peilin Yang

Approved: _____
Xxxx Xxxx, Highest Degree
Chair of the Department of Xxxx

Approved: _____
Xxxx Xxxx, Highest Degree
Dean of the College of Xxxx

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: _____

Xxxx Xxxx, Highest Degree
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: _____

Xxxx Xxxx, Highest Degree
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: _____

Xxxx Xxxx, Highest Degree
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: _____

Xxxx Xxxx, Highest Degree
Member of dissertation committee

# ACKNOWLEDGEMENTS

# 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. There are different types of IR systems targeting to meet different users' information needs. Two IR systems are of the most interest in our works: (1) the evaluation system for IR researchers who are interested in understanding the IR ranking models, (2) the contextual suggestion system that shows interesting venues to end users. In order to make better aforementioned IR systems that are satisfied for both researchers and end users, in our work we explorer the impact of context in two specific long-term IR related tasks – ranking model performance upper bound analysis using unified evaluation system and contextual suggestion.

A long standing problem of IR community is that newly proposed ranking models are implemented and evaluated using various systems of researchers' choices and such choices lead to the fact that different researchers reported different results for the same baseline model. The context – the IR evaluation system used by researcher, is apparently the cause of such problem. This fact casts doubts on the real effectiveness of the proposed ranking models and we argue that it is essential for the IR community to have a unified evaluation system so that ranking models can be compared without considering the details – usually other components of an IR system, e.g. preprocessing of documents or the choice of evaluation metrics. In our work we propose two unified IR evaluation systems to alleviate this problem. VIRLab as our first attempt provides a web based IR evaluation system where the users of the system can implement their own models and get them automatically evaluated. Another system RISE (Reproducible IR System Evaluation) is proposed in order to further improve VIRLab. To the best of our knowledge, RISE is the first ever known system that users of the system can collaborate with each other and thus make the normalization of baseline models much

more easier and thus can be trusted by future researchers. With the help of RISE a large scale reproducibility experiment is deployed and the comprehensive results serve as the the performances reference of most widely used IR models.

Based on the reproducibility experiment results of using RISE we find that the optimum performances of bag-of-terms ranking models do not improve a lot on TREC collections for over 20 years. This introduces another interesting question: does it exist the performance upper bound for bag-of-terms models? To answer this question we extensively investigate the context – the bag-of-terms document representation assumption and the statistics like term frequency and inverted document frequency with which the models usually play. We first find that for single term queries several ranking models can be transformed to a simplified model. We then apply the cost/gain analysis which is commonly used in learning-to-rank (LTR) domain to find the optimum of single term queries for this simplified model. The result shows that although the performances of state-of-the-art ranking models are quit close to the practical optimum there is still some room for improvement.

For contextual suggestion system it is commonly agreed that successful modeling user profile is the key to the solution. We argue that user's preference which is usually treated as a static factor should also be considered as part of the context of this problem. Our first approach is to model the user profile using the venue's category and description from user's activity history. We further improve the method by leveraging the opinions from user's preference to model the user profile. Such methodology utilizes the rich text associated with the users which naturally fits the problem into IR domain. By modeling the candidate suggestions in the similar fashion, the similarities between the candidates and the user profile are used to score the candidates. Experiments on TREC collections and a Yelp data collection confirm the advantage of our method.

For the future work, there are mainly two directions we would like to explore more. The first one is to quantify the impact of the context of unified IR evaluation system. There is no previous work on how much difference does the usage of different retrieval tools bring. We hope to be the first to report on standardizing and quantifying

the impact so that the IR community could be aware of such divergence and can better evaluate the contributions of using various tools. The second one is to provide more sound justification about the boundary theory of ranking model performance. Specifically, we want to extend the current analysis on the single term queries to multiple term queries. We also would like to try other method, e.g. explanatory analysis, to achieve the same goal.

## 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 make the search results of high quality in order to meet the users 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. Generally speaking, there are different types of IR systems targeting to meet different users' information needs. For IR researchers who are interested in understanding the IR ranking models, what they want is the IR system where they can modify the ranking model of the system more easily and quickly, test it, iterate to next round. On the other hand, an IR system could also be mainly for end users. Users of the system do not necessarily know the details of how to design, how to implement the ranking algorithm of the system – they just need to know how to use the system to meet their information needs. One such example is the contextual suggestion system [70–73, 76]. In our work we focus on two IR systems which are both related long-term information retrieval task. The first one is ranking model performance upper bound analysis using unified evaluation system which is mainly for

---

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

IR researchers and the other is contextual suggestion which is mainly for end users. Specifically, we explorer the context and its impact for these two systems.

The word "*Context*" is originally defined as "the set of circumstances or facts that surround a particular event, situation, etc." In our work, context is extensively studied in order to reveal and quantify its impact for different research problems. We will show more details in the following.

## 1.1 Unified IR Evaluation System

The first problem we address is a long standing problem in evaluating the effectiveness of IR system[2]. 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 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. For this problem the unified testing environment is regarded as the context of the evaluation process.

Apparently the context of this problem is the very basis of comparing the effectiveness of ranking models and thus should be carefully treated. Without this context researchers cannot make sound claim about their proposed models. Unfortunately, there is no such environment for the IR community. People continuously report different performances on the same baseline model [75] and this casts doubt on the real effectiveness of the proposed models.

In our work, two systems, namely VIRLab (Virtual IR Lab) [21] and RISE (Reproducible Information retrieval System Evaluation) [75] are proposed in order to

---

[2] There are several aspects in IR system can be evaluated. In our work, we focus on the evaluation of effectiveness of the system. Specifically only the effectiveness of the ranking model is investigated

offer a unified context to the IR community for standardization of comparing ranking models. The uniqueness and the advantage of these two systems is that they offer centralized and controlled IR evaluation systems which facilitate easy but trusted comparison of retrieval models. With the help of these systems (in our case we use 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 TREC collections. 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 all the 16 TREC collections including recently released ClueWeb datasets. To the best of our knowledge, 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.

## 1.2 Boundary Theory of Bag-of-Terms 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 [74, 75] we find that the optimum performances of some models [1, 22, 28, 43, 60, 64, 80] are quite similar on several TREC collections. 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 [20]. The commonalities could be viewed as the context of these ranking models

since they are the theoretical foundation of these retrieval functions. With this context 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 [20]. 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. In our work, we 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 [6,7,18]. 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.

## 1.3 Contextual Suggestion

Another research endeavor of our work is to provide better IR system for end users. This kind of system is different from the previous one since end users do not necessarily know about the implementation or design details of the system – what

they want is how to use the system to meet their information needs. Contextual suggestion is one of such example. The task of contextual suggestion is to recommend interesting venues to the users based on contextual information such as geographic location, temporal information and user's activity history. Traditionally context of this problem is often referred to the physical conditions of the users. In our work we argue that user's long-term, static preferences should also be included in the context as it is the key condition we have to rely on.

There are two necessary steps to tackle the contextual suggestion problem and both of them rely on the context we have defined: (1) identify a set of candidates that satisfy the contextual requirements, e.g., places of interest that are close to a target place; (2) rank the candidate suggestions based on user's interests. User profiling is the key component to effectively rank candidate places with respect to a user's information need and this is the reason we include the user preference history into the context.

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 [69]. The advantage of such approach is the ease of computation and the satisfactory results [15]. 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 [70–73, 76] 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 TREC collections and a self-crawled Yelp collection validate the effectiveness of our method.

## 1.4 Summary

Previous studies in IR rarely separated the context apart from other components of a specific research problem. However, we argue that the context of different IR systems or components should be carefully treated and extensively studied as the impact of the context is big enough to influence the results some IR studies, e.g. the ones aforementioned above.

As of future work, we would further explorer in two directions. The first one is to quantify the impact of the context of unified IR evaluation system. There is no previous work on how much difference does the usage of different retrieval tools bring. We hope to be the first to report on standardizing and quantifying the impact so that the IR community could be aware of such divergence and can better evaluate the contributions of using various tools. The second one is to provide more sound justification about the boundary theory of ranking model performance. Specifically, we want to extend the current analysis on the single term queries to multiple term queries. We also would like to try other method, e.g. explanatory analysis, to achieve the same goal.

The rest of the thesis is organized as follows. First, we discuss related work in Chapter 2. We describe our reproducibility study using VIRLab and RISE in Chapter 3. We explain how we use cost/gain analysis to find the performance upper bound for single term queries in Chapter 4. In Chapter 5, we investigate the problem of contextual suggestion and present our approaches. We then discuss future work in Chapter 6. Finally, we summarize the contributions of the thesis on Chapter 7.

# Chapter 2

# RELATED WORK

In this thesis, we investigate the impact of context in different IR systems. Specifically, two kinds of systems are mainly explored. The first one is the unified evaluation system where the context is the evaluation system itself on which the comparison of different ranking models relies. With the help of the unified evaluation system we tried to provide analytical performance upper bound for bag-of-terms ranking models of which the context here are the statistics used by the models. The other system is the contextual suggestion. The key of solving this problem is to build the user profile based on the user preference history. We include user preference as part of the context of this problem as this is the long term and thus static information need of the user. Extensive work have been done on solving these research questions. We now survey the related work in the literature and discuss the differences with our proposed approaches in detail.

## 2.1 Unified IR Evaluation System

There have been significant efforts on developing various web services for IR evaluation. Lin et al. [39] 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. [25] 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. [26] 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 [35]. 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.

Our developed systems *VIRLab* and *RISE* are closely related to the ideas of Privacy Preserved Evaluation (PPE) [21] and Evaluation as a Service (EaaS) [40,41,59]. VIRLab [21] provides a web service for users to implement retrieval functions. It is mainly designed to facilitate teaching IR models. RISE is also designed as a web service to provide a unified interface for the users to evaluate their models/algorithms. This design enables the system to host the data collections instead of shipping the data collections to researchers, which can ensure the privacy of the collections. Users of RISE can not see the functions implemented by other users. The uniqueness of *RISE* system is that it is specifically designed to facilitate the implementation and evaluation of retrieval functions.

The SIGIR 2015 Workshop on Reproducibility, Inexplicability, and Generalizability of Results (RIGOR) [29] 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. [67] and Muhleisen el. al. [49] 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 [67] and 9 functions 10 collections for [49]) are not as large as what we studied in this paper.

Compared with the previous studies, our work is different in the following two aspects. First, the *RISE* system is specifically designed for the reproducibility study of retrieval models. It hides details about collection processing and evaluation, and enables users to focus on only the implementation of retrieval models. Due to 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.

## 2.2 Boundary Theory of Bag-of-Terms Models

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 [20] 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 [6,7] and the LambdaRank [6,18] 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 [78]. 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. [66] 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.

## 2.3 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 [12–15]. 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 [13,15]. After that, the task is to retrieve useful suggestions based on user preferences. Most participants formulated the task as a content-based recommendation problem [30, 32, 37, 38, 47, 58, 61, 68, 69, 72, 77]. 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 [30, 32, 69]. A few studies also explored the use of category information for user profiling and candidate ranking. For example, Li and Alonso [38] utilized the accumulative category scores to model both user and candidate profiles, and then use the full range cosine similarity between the

10

two profiles for candidate ranking. Li et al. [37] 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. [47] 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. [77] proposed to recommend the candidates which are proportional to the number of example suggestions in each category. Koolen et al. [34] applied a similar method with a major modification of retrieving the category information from Wikitravel[1].

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.

### 2.3.1 Recommendation Systems

The problem of contextual suggestion is also similar to collaborative filtering [65]. 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 [63]. In order to solve the data sparsity problem, reviews were incorporated to improve the performance. Hariri et al. [27] 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 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. [31] 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. [50] applied random walk based on latent space models and computed a variety of

---

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

11

similarity criteria with venue's visit frequencies on the location based social newtowkr. Bao et al [3] 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 [27, 36, 55, 57, 62]. Raghavan et al. [57] 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. [36] extended this study and analyzed the review texts to get the intent, features and the ratings for each feature. Qumsiyeh and Ng [55] 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.

### 2.3.2  Text Summarization

The summary generation of our work is related to automatic text summarization. Automatic text summarization has been well studied for traditional documents such as scientific documents and news articles [56]. In particular, previous work has studied various problems in this area including extractive summarization, abstractive

summarization, single-document summarization and multiple-document summarization [11]. More recently, there have been effort on opinion mining and summarization [2, 9, 16, 17, 19, 33, 45, 46, 52–54]. Most of them involve in the finer partition of the reviews and polarity judging of each partition. Common strategies include part-of-speech analysis, negation identification and etc. Unlike the previous effort, we focus on generating a *personalized* summary for a suggestion. Since the information about the suggestion is scattered in many places, including description, web sites and reviews, the summarization needs to synthesize the information from these heterogeneous information sources. Instead of extracting the information from a single source, we try to leverage one information source to guide the extractive summarization process in other sources and then assemble all the extracted summaries together into a *structural* way. Another main difference of our work from previous studies is to utilize the user profile to generate personalized summaries.

# Chapter 3

# REPRODUCIBILITY STUDY USING UNIFIED IR EVALUATION SYSTEM

Developing effective information retrieval models has been a long standing challenge in Information Retrieval (IR), and significant progresses have been made over the years. With the increasing number of developed retrieval functions and the release of new data collections, it becomes more difficult, if not impossible, to compare a new retrieval function with all existing retrieval functions over all available data collections. The context – the evaluation system is crucial to solving this problem. To tackle this problem, we construct the unified IR evaluation systems that aim to improve the reproducibility of IR research and facilitate the evaluation and comparison of retrieval functions. With the developed platform, 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, the developed platform (i.e., *RISE*) is made publicly available so that any IR researchers would be able to utilize it to evaluate other retrieval functions.

## 3.1 VIRLab

Existing IR toolkits do not offer an easy solution for users to implement, evaluate and analyze new retrieval models, which often pose unnecessary extra burdens to the users. For example, users need to learn how to access various term or document statistic

information from the index, and read through source codes or API documentations to figure out how to implement a new retrieval function. After implementing the function, the users often need to write their own scripts to evaluate and analyze the search results over multiple collections. This kind of process adds unnecessary burden to the users and could discourage them from experimenting with more functions over more collections. Naturally, it is also very challenging to use such toolkits for a course assignment.

In this section we describe our efforts on developing a web-based tool for IR students or researchers to study retrieval functions in a more interactive and cost-effective way. We will demonstrate that the developed system, i.e,. Virtual IR Lab (VIRLab), can offer the following new 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.
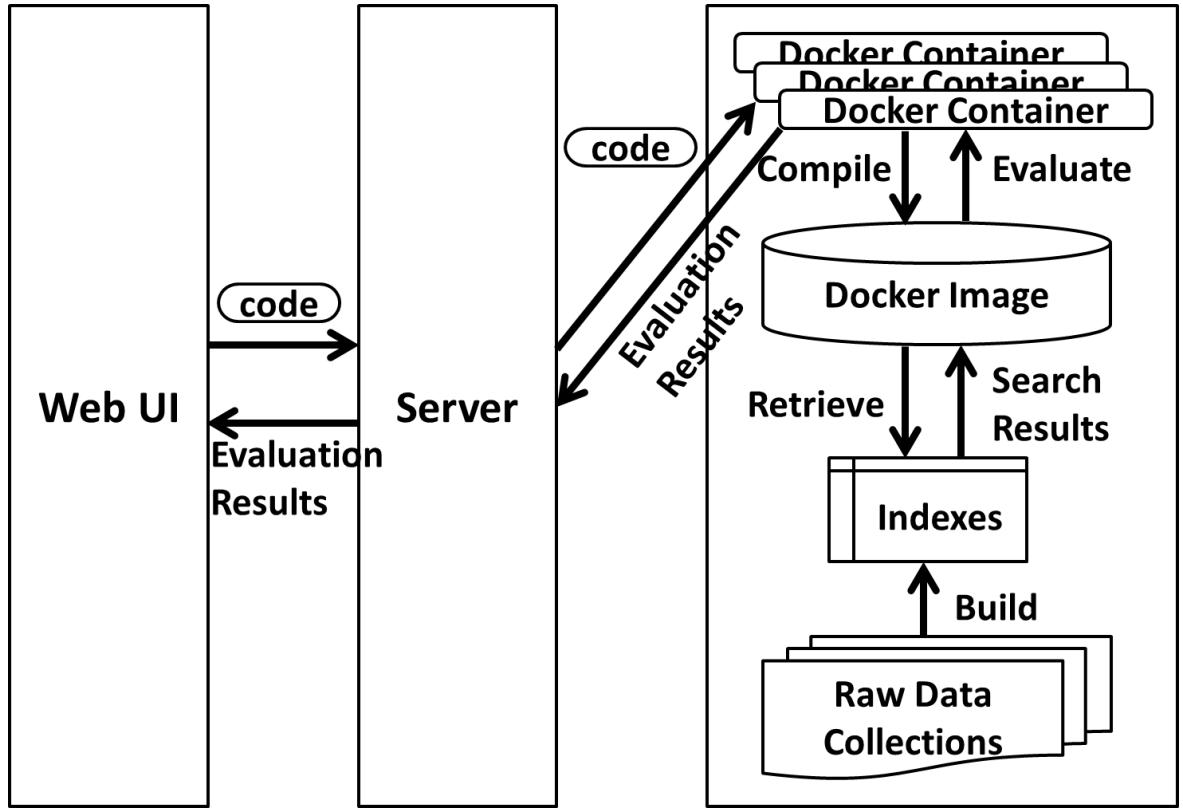
Figure 3.1: System Architecture

## 3.2    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 3.1 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.

---

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

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. 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

Table 3.1: 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 | $\sum_{t\in q} \frac{(k_3+1)\cdot c_t^q}{k_3+c_t^q} \cdot \frac{(k_1+1)\cdot tfn}{k_1+tfn} \cdot ln\left(\frac{N-N_t+0.5}{N_t+0.5}\right)$ <br> $tfn = \frac{c_t^d+\mu\cdot\frac{F_t}{|C|}}{l_d+\mu} \cdot \mu$ |
| 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 | $\sum_{t\in q} \left(1 + ln(1 + ln(c_t^d))\right) \cdot \frac{L+s}{L+s\cdot l_d} \cdot \left(\frac{N+1}{N_t}\right)^k$ |
| F1LOG | $\sum_{t\in q} \left(1 + ln(1 + ln(c_t^d))\right) \cdot \frac{L+s}{L+s\cdot l_d} \cdot ln\left(\frac{N+1}{N_t}\right)$ |
| 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 | $\sum_{t\in q} \left[\left[\omega \cdot f\left(\frac{c_t^d}{l_d/c_d}\right) + (1-\omega)\cdot f\left(c_t^d \cdot log_2\left(1+\frac{L}{l_d}\right)\right)\right] \cdot \left[ln\left(\frac{N+1}{N_t}\right) \cdot f\left(\frac{F_t}{N_t}\right)\right]\right]$ <br> $\omega = \frac{2}{1+log_2(1+|q|)},\ f(x) = \frac{x}{1+x}$ |

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.

## 3.3 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 [51, 64], the classic probabilistic models [60], the language modeling approaches [79, 80], the divergence from randomness

models [1], the axiomatic models [22,43], and the information theory based models [10].
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 $t$

- $|C|$: 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 4.1 and Table 3.2.

### 3.3.1   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 [60], 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 [22] 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

19

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 [28]. 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 [43] 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.

### 3.3.2   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 [64]. 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 [22]. 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 [51]. 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.

### 3.3.3 Language modeling approaches

Dirichlet prior method, denoted as **DIR**, is one of the representative retrieval functions derived using the language modeling approaches [80]. 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 [79]. 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 [48]. 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 [22] are denoted as **F3EXP** and **F3LOG**. The variant derived based on the lower bound term frequency normalization [43] is denoted as **DIR+**.

### 3.3.4 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 [28]. 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 [43]. It is denoted as **PL2+**.

### 3.3.5 Information-based Models

A family of information-based models was proposed for ad hoc IR [10]. 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**.

## 3.4 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.

### 3.4.1 Reproducibility study

#### 3.4.1.1 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 3.3.

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].

### 3.4.1.2 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 3.3. 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 3.4 and Table 3.5 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

---

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

report the performances on the collections that we select. Instead, we use what were reported in [28,43] 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 3.6. 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 3.7. 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 [49], using different toolkits could lead to different evaluation results.

### 3.4.2 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

---

[3] http://terrier.org/

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 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.

---

4 http://lemurproject.org/clueweb09/

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

Figure 3.2: Optimal Performances on ClueWeb Collections
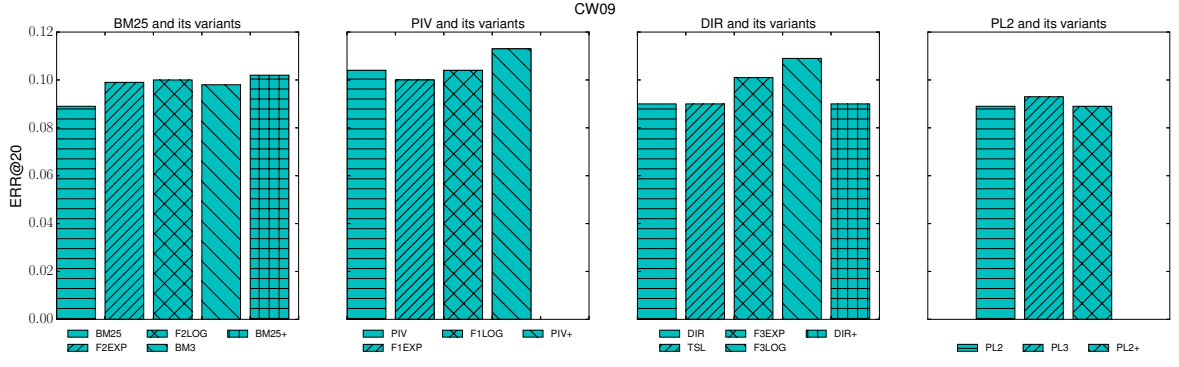
(a) Performances of selected models on CW09
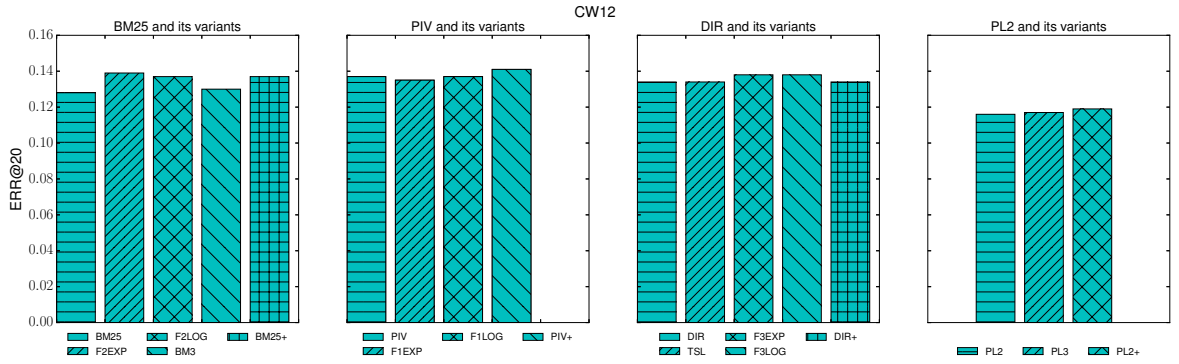
(b) Performances of selected models on CW12

Figure 3.2 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.

### 3.4.3 Summary

To serve as a future reference, we summarize the optimal performance of all the retrieval functions over all the data sets in Table 3.8. 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 3.9.

The optimal performances for the selected retrieval models on all collections are shown in Table 3.8. To the best of our knowledge this is the first time of reporting such large scale and comprehensive performances of retrieval models.

### 3.5 Conclusions

This paper describes our efforts on building the Reproducible Information retrieval System Evaluation (*RISE*) platform. *RISE* is a Web service that facilitates the implementation and evaluation of IR models. In particular, it can serve as an implementation repository of retrieval functions. Users can not only submit their own implementations but also view the implementations submitted by other users. With such an implementation repository, the *RISE* can also facilitate the evaluation of existing retrieval functions over new data collections. 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[6].

---

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

Table 3.2: Retrieval functions that are reproduced in our study (Part 2)

| | |
|---|---|
| **Language modeling approaches** | |
| DIR | $\sum_{t\in q} ln\left(\frac{c_t^d + \mu\cdot\frac{F_t}{|C|}}{l_d+\mu}\right)$ |
| TSL | $\sum_{t\in q}\left((1-\lambda)\cdot\frac{c_t^d+\mu\cdot\frac{F_t}{|C|}}{l_d+\mu} + \lambda\cdot\frac{F_t}{|C|}\right)$ |
| BLM | $\sum_{t\in q}\frac{c_t^d+\mu\cdot\frac{F_t}{|C|}}{l_d+\frac{|C|}{F_t}+\mu-2}$ |
| F3EXP | $\sum_{t\in q}\left(1+ln(1+ln(c_t^d))\right)\cdot\left(\frac{N+1}{N_t}\right)^k - \frac{(l_d-|q|)\cdot|q|\cdot s}{L}$ |
| F3LOG | $\sum_{t\in q}\left(1+ln(1+ln(c_t^d))\right)\cdot ln\left(\frac{N+1}{N_t}\right) - \frac{(l_d-|q|)\cdot|q|\cdot s}{L}$ |
| DIR+ | $\sum_{t\in q}\left[ln\left(1+\frac{c_t^d}{\mu\cdot\frac{F_t}{|C|}}\right)+ln\left(1+\frac{\delta}{\mu\cdot\frac{F_t}{|C|}}\right)\right] + |q|\cdot ln\frac{\mu}{l_d+\mu}$ |
| **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 | $\sum_{t\in q}\frac{tfn\cdot log_2(tfn\cdot\lambda)+log_2 e\cdot(\frac{1}{\lambda}-tfn)+0.5\cdot log_2(2\pi\cdot tfn)}{tfn+1}$ $$tfn = \frac{c_t^d+\mu\cdot\frac{F_t}{|C|}}{l_d+\mu}\cdot\mu$$ |
| 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 | $\sum_{t\in q}-ln\left(\frac{\lambda_t^{\frac{n_t^d}{n_t^d+1}}-\lambda_t}{1-\lambda_t}\right)$ $$\lambda_t = \frac{F_t}{N} \text{ and } n_t^d = c_t^d + ln\left(1+c\cdot\frac{L}{l_d}\right)$$ |
| LGD | $\sum_{t\in q}-ln\left(\frac{\lambda_t}{n_t^d+\lambda_t}\right)$ $\lambda_t$ and $n_t^d$ as shown above |

28

Table 3.3: Data collections used for the reproducibility study

| | Topics | Doc. collection | #documents | avdl |
|---|---|---|---|---|
| ad hoc task at TREC-1 | 51-100 | **disk1&2** | 741,856 | 412.89 |
| ad hoc task at TREC-2 | 101-150 | | | |
| ad hoc task at TREC-3 | 151-200 | | | |
| ad hoc task at TREC-6 | 301-350 | **disk4&5** | 528,155 | 467.553 |
| ad hoc task at TREC-7 | 351-400 | | | |
| 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 | **GOV2** | 25,205,179 | 937.252 |
| terabyte track at TREC 2005 | 751-800 | | | |
| terabyte track at TREC 2006 | 801-850 | | | |

Table 3.4: 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 3.5: 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 3.6: 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 3.7: 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 3.8: Optimal MAP/ERR@20 for all collections. * indicates the model is significantly 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$^\dagger$ |
| F2LOG | 0.231* | 0.252$^\dagger$ | 0.302 | 0.297 | 0.100* | 0.137 |
| BM3 | 0.234* | 0.241 | 0.296 | 0.283 | 0.098* | 0.130 |
| BM25+ | 0.235*$^\dagger$ | 0.249 | 0.318$^\dagger$ | 0.301$^\dagger$ | 0.102*$^\dagger$ | 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*$^\dagger$ | 0.239*$^\ddagger$ | 0.299* | 0.265* | 0.113$^\dagger$ | 0.141$^\dagger$ |
| NTFIDF | 0.205* | 0.213 | 0.307*$^\dagger$ | 0.296*$^\ddagger$ | 0.097 | 0.129 |
| Language Modeling Approaches | | | | | | |
| DIR | 0.227 | 0.252$^\dagger$ | 0.312 | 0.299 | 0.090 | 0.134 |
| BLM | 0.208 | 0.233 | 0.314$^\dagger$ | 0.222 | 0.072 | 0.113 |
| TSL | 0.228$^\dagger$ | 0.252 | 0.312 | 0.300$^\dagger$ | 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*$^\dagger$ | 0.138$^\dagger$ |
| 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$^\dagger$ | 0.089 | 0.116 |
| PL3 | 0.228$^\dagger$ | 0.241 | 0.290 | 0.269 | 0.093$^\dagger$ | 0.117 |
| PL2+ | 0.214 | 0.255*$^\ddagger$ | 0.328*$^\ddagger$ | 0.301 | 0.089* | 0.119*$^\dagger$ |
| Information-based Models | | | | | | |
| LGD | 0.215$^\dagger$ | 0.251$^\dagger$ | 0.320$^\dagger$ | 0.300$^\dagger$ | 0.086 | 0.131$^\dagger$ |
| SPL | 0.213 | 0.251 | 0.313 | 0.299 | 0.093$^\dagger$ | 0.130 |

Table 3.9: 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 | $\mu \in [500, 5000]$ | 500 |
| | $\lambda \in [0, 1]$ | 0.1 |
| PL2 | $c \in [0.5] \cup [1, 25]$ | 1 |
| BM3, PL3 | $c \in [0.5] \cup [0.75] \cup [1, 9]$ | 1 |
| | $\mu \in [500, 5000]$ | 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 4

# BOUNDARY THEORY OF IR RANKING MODELS

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.

## 4.1 Performance Bound Analysis

### 4.1.1 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.

Table 4.1: Instantiations of the general retrieval form

| Retrieval Functions | $g(\cdot)$ | $\alpha$ | $c_1$ | $\gamma$ | $\beta$ | $c_2$ |
|---|---|---|---|---|---|---|
| DIR | 1 | 1 | $\mu \cdot p(t\|C)$ | 0 | 1 | $\mu$ |
| 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 | $\mu \cdot p(t\|C)$ | $\mu$ | $k_1$ | $k_1 \cdot \mu + \mu^2 \cdot p(t\|C)$ |
| DIR+ | 1 | $\mu \cdot p(t\|C)+\delta$ | $\mu^2 \cdot p^2(t\|C)+\delta \cdot \mu \cdot p(t\|C)$ | 0 | $\mu \cdot p(t\|C)$ | $\mu^2 \cdot p(t\|C)$ |

Let us start with a specific example. Dirichlet prior function is one of the representative functions derived using language modeling approaches [80], and is shown as follows:

$$f(Q,d) = \sum_{t \in Q} \ln\left(\frac{c(t,d) + \mu \cdot p(t|C)}{|d| + \mu}\right), \tag{4.1}$$

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

$$f(\{t\},d) = \frac{c(t,d) + \mu \cdot p(t|C)}{|d| + \mu} \tag{4.2}$$

Note the natural logarithm function in Equation (4.1) 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:

$$f(t,d) = \frac{c(t,d) + c_1}{|d| + c_2}. \tag{4.3}$$

Similarly, Okapi BM25 [60] can be simplified to:

$$\begin{aligned} f(t,d) &= \frac{(k_1+1) \cdot c(t,d)}{c(t,d) + k_1 \cdot (1-b+b \cdot |d|/avdl)} \\ &= \frac{\alpha \cdot c(t,d)}{c(t,d) + \beta \cdot |d| + c_2}, \end{aligned} \tag{4.4}$$

where $\alpha$ 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) [64] can also be simplified to:

$$f(t, d) = \frac{1 + ln(1 + ln(c(t, d)))}{(1 - s + s \cdot |d|/avdl)}$$
$$= \frac{g(c(t, d))}{(\beta \cdot |d| + c_2)}, \tag{4.5}$$

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. (4.3), Eq. (4.4) and Eq. (4.5)) can be generalized as the following form:

$$F(c(t, d), |d|) = \frac{\alpha \cdot g(c(t, d)) + c_1}{\gamma \cdot c(t, d) + \beta \cdot |d| + c_2}, \tag{4.6}$$

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. (4.6) 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 [22], (2) BM3 derived from the Dirichlet Priors for term frequency normalization model [28], and (3) BM25+, DIR+, PIV+ derived from the lower bounding term frequency normalization models [43]. Table 4.1 summarizes the instantiations for each of the retrieval functions.

### 4.1.2 Upper Bound Estimation for MAP

Given the general form as shown in Equation (4.6), 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. (4.6), 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 paper). 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 [6,7], we define the cost function as the pairwise cross-entropy cost applied to the logistic of the difference of the relevance scores:

$$C_{ij} = \frac{1}{2}(1 - S_{ij})\sigma(s_i - s_j) + \log(1 + e^{-\sigma(s_i - s_j)}) \tag{4.7}$$

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:

$$\frac{\partial C_{ij}}{\partial s_i} = \sigma\left(\frac{1}{2}(1 - S_{ij}) - \frac{1}{1 + e^{\sigma(s_i - s_j)}}\right) = -\frac{\partial C_{ij}}{\partial s_j} \tag{4.8}$$

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. (4.8) is then simplified as:

$$\frac{\partial C_{ij}}{\partial s_i} = \frac{-\sigma}{1 + e^{\sigma(s_i - s_j)}} \tag{4.9}$$

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

$$p_k \rightarrow p_k - \eta \frac{\partial C}{\partial p_k} = p_k - \eta \left( \frac{\partial C}{\partial s_i} \frac{\partial s_i}{\partial p_k} + \frac{\partial C}{\partial s_j} \frac{\partial s_j}{\partial p_k} \right) \tag{4.10}$$

Unfortunately, the cost defined in Eq. (4.9) is actually the "optimization" cost instead of the target cost (the actual cost) [7] 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 [78]. To overcome this we modify Eq. (4.9) 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 $\lambda$ (we change cost $C$ to $\lambda$ and $\lambda$ is the gain instead of cost) can be written as:

$$\lambda_{ij} = \frac{\sigma}{1 + e^{\sigma(s_i - s_j)}} \frac{1}{|R|} \left( \left| \frac{n}{r_j} - \frac{m}{r_i} \right| + \sum_{k=r_j+1}^{r_i-1} \frac{I(k)}{k} \right) \tag{4.11}$$

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 $r_j$; $I(k) = 1$ if the document at $k$th 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 $\lambda$ 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 [18].

## 4.2  Experiments

### 4.2.1  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

Table 4.2: 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 |

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 4.2.

### 4.2.2 Experiment Setup

We tested both using the cost function only and using the cost function together with $\lambda$ component of MAP. The results are very close and the cost with $\lambda$ 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. (4.6):

- **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 $\frac{c(t,d)+c_1}{|d|+c_2}$

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

For other possible format of Eq. (4.6) 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 $\eta$ 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 $\alpha, \beta, c_1, c_2$ 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.

Table 4.3: 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** | $\mathbf{0.3255^{\dagger}}$ |

### 4.2.3 Results

Table 4.3 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 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. (4.6).

---

[1] Actually they are possibly covered by Eq. (4.6). But if we choose wide spectrum of the starting points then they are covered by large chance.

| Model | Paras | disk12 | Robust04 | WT2G | GOV2 |
|-------|-------|--------|----------|------|------|
| $\text{DIR}^U$ | $\mu$ | 4.66e3 | 3.54e7 | 1.43e6 | 0 |
| $\text{TFDL1}^U$ | $\frac{c_1}{c_2}$ | 2.49e-3 | 1.0 | 6.87e-5 | 6.0e-1 |
| $\text{TFDL2}^U$ | $\frac{c_1}{c_2}$ | 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 |

### 4.2.4   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 4.4 lists those parameters. As we can see, for $\text{DIR}^U$ the optimal parameters $\mu$ obtained for Robust04 and WT2G are much larger than $10^3$ which is suggested value by the original authors of DIR [80]. For $\text{TFDL1}^U$ 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 $\text{TFDL2}^U$ both $\frac{c_1}{c_2}$ and $\frac{\alpha}{\beta}$ are reported. We find that $\alpha$ is in several magnitude levels smaller than $\beta$. 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.

### 4.3   Conclusions

We have applied cost/gain analysis to the performance upper bound of single term queries for TREC collections. The found upper bounds of MAP provide sound foundation of potentially better performed ranking models in the future. Moreover, the parameters that lead to the local optimums provide more insight about how the future models could better incorporate proper statistics/signals.

# Chapter 5

# 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.

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.

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.

## 5.1 Problem Formulation

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.

## 5.2 Category and Description based User Profile Modeling

### 5.2.1 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:

$$
\begin{aligned}
\text{S(U, CS)} &= \varphi \times \text{SIM}(\mathcal{U}_+(U), CS) + (1 - \varphi) \times \text{SIM}(\mathcal{U}_-(U), CS) && (5.1) \\
&= \varphi \times \frac{\sum_{e \in \mathcal{U}_+(U)} \text{SIM}(e, CS)}{|\mathcal{U}_+(U)|} + (1 - \varphi) \times \frac{\sum_{e \in \mathcal{U}_+(U)} \text{SIM}(e, CS)}{|\mathcal{U}_+(U)|} && (5.2)
\end{aligned}
$$

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 that the user likes. When $\varphi = 0$, the highly ranked suggestions would be those different from the suggestions that the user dislikes. $\text{SIM}(\mathcal{U}_+(U), CS)$ 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., $\text{SIM}(e, CS)$, where $e$ is an example from the user profile. We explore the following two types of information to compute $\text{SIM}(e, CS)$: (1) the category of a place; and (2) the description of a place.

### 5.2.1.1 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.

Table 5.1: 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 |
| ... | ... |

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

$$\text{SIM}_{\mathcal{C}}(e, CS) = \frac{\sum_{c_i \in \mathcal{C}(e)} \sum_{c_j \in \mathcal{C}(C)} \frac{|Intersection(c_i, c_j)|}{max(|c_i|, |c_j|)}}{|\mathcal{C}(e)| \times |\mathcal{C}(C)|} \qquad (5.3)$$

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 5.1 shows some example categories of the suggestions.

### 5.2.1.2  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 [22] since it has been shown to be effective for long queries [22]. So, we have

$$\text{F2EXP}(a, b) = \sum_{t \in a \cap b} \frac{c(t, b)}{c(t, b) + 0.5 + \frac{0.5 \cdot |b|}{avdl} \cdot \left(\frac{N+1}{df(t)}\right)^{0.35}} \qquad (5.4)$$

Thus, we compute the similarity scores as follows:

$$\text{SIM}_{\mathcal{D}}(e, CS) = \text{F2EXP}(DES(e), DES(CS)) \qquad (5.5)$$

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

## 5.3 Opinion-based User Profile Modeling

### 5.3.1 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 [4, 30, 34, 58, 69–71]. 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 [61, 69, 77]. 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 5.5.2.1.

We now motivate the opinion-based user modeling through an example as shown in Figure 5.1. 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:

$$\mathcal{U}_+(U) = \bigcup_{\forall i, R(U, ES_i) = POS} REP_+(ES_i), \tag{5.6}$$

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 5.3.2).

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

$$\mathcal{U}_-(U) = \bigcup_{\forall i, R(U, ES_i) = NEG} REP_-(ES_i), \qquad (5.7)$$

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

### 5.3.2 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 [24], 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 5.2 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.

### 5.3.3 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.

### 5.3.3.1 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 5.3. 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., $\mathrm{SIM}(\mathcal{U}_+(U), REP_+(CS))$ and $\mathrm{SIM}(\mathcal{U}_-(U), REP_-(CS))$. 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., $\mathrm{SIM}(\mathcal{U}_+(U), REP_-(CS))$ and $\mathrm{SIM}(\mathcal{U}_-(U), REP_+(CS))$.

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

$$
\begin{aligned}
S(U, CS) = {} & \alpha \times \mathrm{SIM}(\mathcal{U}_+(U), REP_+(CS)) - \beta \times \mathrm{SIM}(\mathcal{U}_+(U), REP_-(CS)) \\
& - \gamma \times \mathrm{SIM}(\mathcal{U}_-(U), REP_+(CS)) + \eta \times \mathrm{SIM}(\mathcal{U}_-(U), REP_-(CS))
\end{aligned}
\tag{5.8}
$$

where $\alpha$, $\beta$, $\gamma$ and $\eta$ 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 [22] since it has been shown to be effective for long queries [22]. So, we have

$$
\mathrm{SIM}(a, b) = \sum_{t \in a \cap b} \frac{c(t, b)}{c(t, b) + 0.5 + \frac{0.5 \cdot |b|}{avdl} \cdot \left(\frac{N+1}{df(t)}\right)^{0.35}}
\tag{5.9}
$$

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.

### 5.3.3.2 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 [42, 44].

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 5.3.3.1). 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 [23]. 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 [5]. 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.

## 5.4 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 [13].

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 ..."

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:

---

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

53

- **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 PEO-PLE: 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."*

## 5.5 Experiments

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

### 5.5.1 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

Table 5.2: 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]                |

track [15]; and (2) a data set crawled from Yelp[2].

- **TREC data set:** The TREC Contextual Suggestion Track [15] 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 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 5.2.

- **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.

---

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

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

### 5.5.2 Experiments on Candidate Suggestion Ranking

#### 5.5.2.1 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 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 5.5.2.3.

As discussed in Section 5.5.1, 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 5.6. 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) [8]. 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.

#### 5.5.2.2 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 5.3. 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 5.3.3. 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 5.4 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.

### 5.5.2.3  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 5.4. 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 5.5 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 5.6. 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.

### 5.5.3   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 [14]. To compare the results of

the two methods, we develop an annotation interface as shown in Figure 5.5. 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 5.7. 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 5.6. 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 5.8 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 5.9 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.

## 5.6 Conclusions

One of the important challenging in mobile search is to return satisfying results based on not only user preference history but also contextual information. Our work focuses on user profiling for contextual suggestion. In particular, we propose to use opinions for both candidate suggestion ranking as well as suggestion summary generation.

For candidate suggestion ranking task, category, description and opinions are used to build the enriched user profile and the candidate suggestion profile. For opinion based user profile several opinion representations including the full review text and part of the review text have been explored to model the user profile. Various ranking methods including linear interpolation and learning to rank have been investigated to compute the ranking scores. The effectiveness of our methods have been tested on standard TREC collections and a self-crawled Yelp collection for both P@5 and ERR@20. The more detailed analysis showed that using NR as the representation of the opinion in general performs better than other opinion representations. Although there is no obvious difference between the optimal performance of linear interpolation and learning to rank, MART performs better than other learning to rank methods - Lambda MART and Linear Regression. We further did more in-depth analysis on how size of the user profile can affect the performance. The results showed that using as

61

few as 10% of the user preferences to build the user profile still leads to promising performance.

Furthermore, we also propose to construct a structured summary by leveraging information from multiple resources. Experiment results show that the proposed method is more effective than the baseline method, and some components in the summary are more useful than the others.

| | 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 5.1: 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.

---

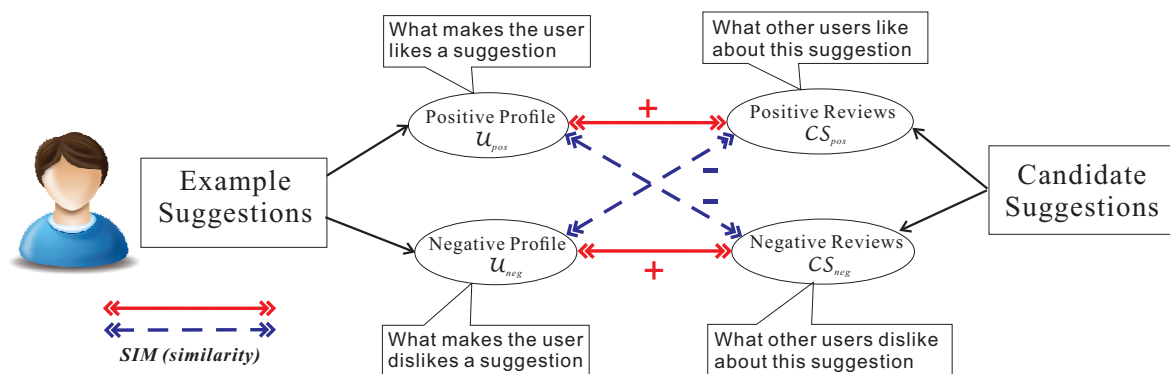Figure 5.2: An example results of different opinion-based representations



Figure 5.3: The linear interpolation method

64

Table 5.3: 5-fold cross validation results using linear interpolation method. $*$ (or $\dagger$) 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*† | 0.68*† |
| | SR | 0.80*† | 0.66*† |
| | NR | 0.81*† | 0.66*† |
| | RS | 0.81*† | 0.67*† |
| *CS2013* | category | 0.66 | 0.68 |
| | description | 0.65 | 0.65 |
| | FR | 0.72*† | 0.70*† |
| | SR | 0.71*† | 0.69*† |
| | NR | 0.71*† | 0.70*† |
| | RS | 0.69*† | 0.68*† |
| *CS2014* | category | 0.72 | 0.74 |
| | description | 0.71 | 0.74 |
| | FR | 0.73*† | 0.76*† |
| | SR | 0.71 | 0.77*† |
| | NR | 0.75*† | 0.78*† |
| | RS | 0.75*† | 0.75*† |
| 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 5.4: 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*† | 0.80*† | 0.52† | 0.63*† | 0.64† |
| | SR | 0.64 | 0.75*† | 0.73† | 0.47 | 0.63*† | 0.56 |
| | NR | 0.64 | 0.74*† | 0.75† | 0.47 | 0.63*† | 0.61† |
| | RS | 0.61 | 0.80*† | 0.76*† | 0.45 | 0.67*† | 0.63† |
| 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† | 0.69*† | 0.72*† | 0.63† | 0.68*† | 0.70*† |
| | SR | 0.65† | 0.69*† | 0.71*† | 0.57 | 0.68*† | 0.69*† |
| | NR | 0.65† | 0.70*† | 0.71*† | 0.64† | 0.68*† | 0.70*† |
| | RS | 0.65† | 0.69*† | 0.71*† | 0.59 | 0.70*† | 0.70*† |
| 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*† | 0.76*† | 0.66 | 0.76*† | 0.79*† |
| | SR | 0.62 | 0.70*† | 0.75*† | 0.60 | 0.72*† | 0.78*† |
| | NR | 0.67 | 0.73*† | 0.75*† | 0.68 | 0.77*† | 0.79*† |
| | RS | 0.66 | 0.73*† | 0.74*† | 0.63 | 0.76*† | 0.79*† |
| 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* |
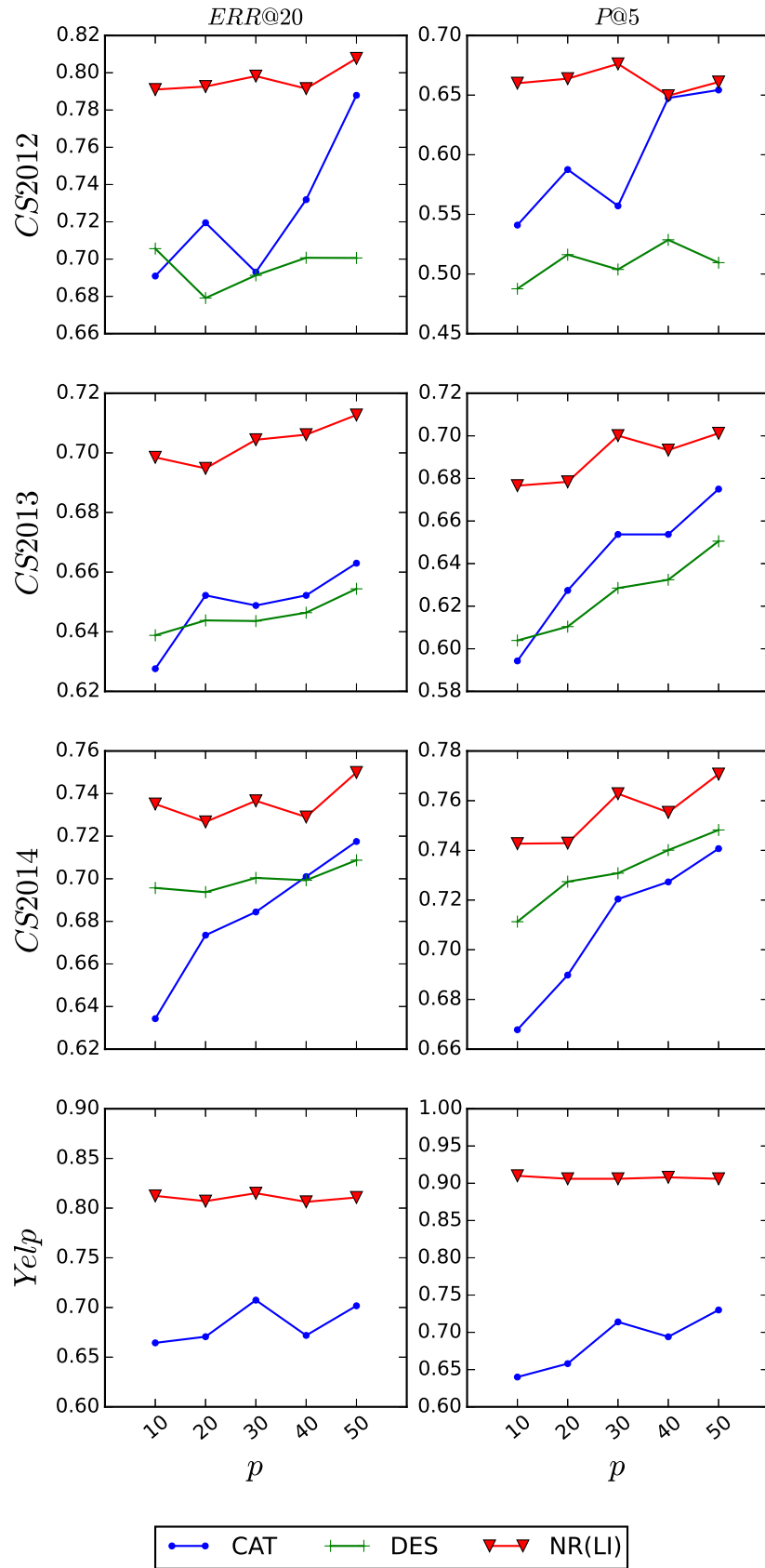
Figure 5.4: The performance of using less data to build user profile

Table 5.5: 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 5.6: KL divergence between positive user profile (id:918) and positive candidate profile (id:107)

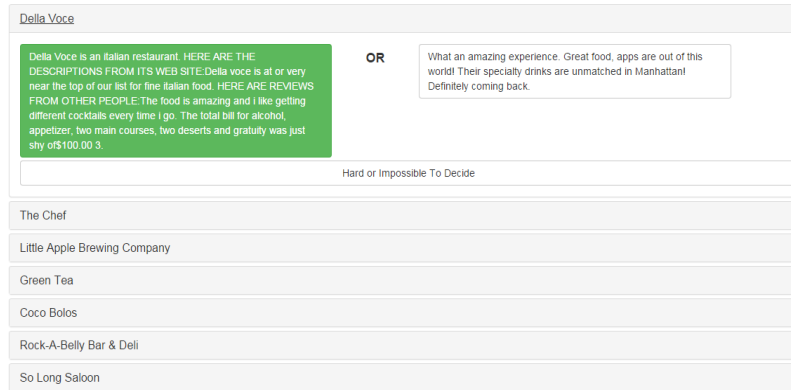| Representations | KL Div. | Ranking |
|---|---|---|
| NR | 1.54 | 7 |
| FR | 0.61 | 2 |
| SR | 1.40 | 2 |
| RS | 0.95 | 5 |

Figure 5.5: Screen shot of the web-based annotation system to compare two summary generation methods

Table 5.7: Comparison of results summarization methods

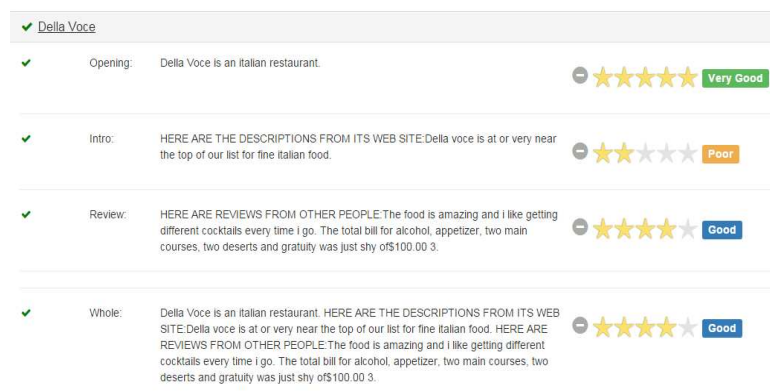| Overlapped Suggestions | | |
|---|---|---|
| | Annotator#1 | Annotator#2 |
| Our method is better than the baseline. | 71% | 86% |
| Our method is worse than the baseline. | 20% | 11% |
| Hard or Impossible to decide | 9% | 4% |
| Non-Overlapped Suggestions | | |
| | Annotator#1 | Annotator#2 |
| Our 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 5.6: Screen shot of the web-based annotation system to evaluate the effectiveness of components

69

Table 5.8: 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 5.9: 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 6

# FUTURE WORK

# 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] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] 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.

[9] Hsinchun Chen and David Zimbra. Ai and opinion mining. *IEEE Intelligent Systems*, 25(3):74–80, May 2010.

[10] 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.

[11] Dipanjan Das and Andre F. T. Martins. A survey on automatic text summarization. 2007.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] Lipika Dey and SK Mirajul Haque. Opinion mining from noisy text data. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(3):205–226, 2009.

[17] Xiaowen Ding and Bing Liu. The utility of linguistic rules in opinion mining. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 811–812. ACM, 2007.

[18] 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.

[19] Andrea Esuli. Automatic generation of lexical resources for opinion mining: models, algorithms and applications. *SIGIR Forum*, 42(2):105–106, 2008.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

[32] Ming Jiang and Daqing He. Pitt at trec 2013 contextual suggestion track. In *Proceedings of TREC'13*, 2013.

[33] Kevin Knight and Daniel Marcu. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107, 2002.

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

[35] 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.

[36] 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.

[37] 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.

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

[39] 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.

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

[41] 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.

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

[43] 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.

[44] 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.

[45] Inderjeet Mani. *Automatic summarization*, volume 3. 2001.

[46] Inderjeet Mani and Mark T Maybury. *Advances in automatic text summarization*. 1999.

[47] 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.

[48] 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.

[49] 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 &#38; Development in Information Retrieval*, SIGIR '14, pages 863–866, New York, NY, USA, 2014. ACM.

[50] 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.

[51] 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.

[52] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010.

[53] Bo Pang and Lillian Lee. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd*

*Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004.

[54] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January 2008.

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

[56] D. Radev, H. Jing, and M. Budzikowska. Introduction to the special issue on summarization. *Computational Linguistics*, 28(4):399.

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

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

[59] 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.

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

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

[62] Jose San Pedro, Tom Yeh, and Nuria Oliver. Leveraging user comments for aesthetic aware image search reranking. In *Proceedings of WWW'12*, 2012.

[63] 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.

[64] 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.

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

[66] 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.

[67] 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.

[68] Di Xu and Jamie Callan. Modelling psychological needs for user-dependent contextual suggestion. In *Proceedings of TREC'14*, 2014.

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

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

[71] 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.

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

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

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

[80] 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.