Fusion and Diversification in Information Retrieval

Shangsong Liang

Fusion and Diversification in Information Retrieval

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.dr. D.C. van den Boom
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op maandag 15 december 2014, te 14:00 uur

door

Shangsong Liang

geboren te Guangxi, China

Promotiecommissie

Promotor:

Prof. dr. M. de Rijke

Overige leden:

Prof. dr. X. Cheng
Prof. dr. C. de Laat
Prof. dr. A. P. de Vries
Dr. C. Monz

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

SIKS Dissertation Series No. 2014-47
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

This research was supported by the China Scholarship Council and the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 288024 (LiMoSINe).

Cover by David Graus
Printed by Off Page, Amsterdam

ISBN: 978-94-6182-522-3

To the unseen heroes,
including you.

I have been studying at the University of Amsterdam as a Ph.D. student since September 8, 2011. Without the help of many people around me, the thesis definitely would not have come about. Here, I would like to say *Thank You!* to everyone who was somehow involved, especially to the following people:

Maarten,
for supervising me on the way to obtaining my Ph.D. degree.

The committee members,
for their valuable input and comments.

Bouke, Edgar, Ilya, Jiyin, Katja, Manos, Marc, and Wouter,
for sharing their research experience.

Aleksandr, Alexey, Anne, Artem, Daan, Damien, David, Fei, Hendra, Hendrike, Masrour, Richard, Ridho, Tom, Xinyi, Yaser, and Zhaochun,
for discussing ideas.

All other current and former ILPSers,
for working together and sharing daily life.

Finally, my parents,
for their love and support for all these years.

Shangsong Liang
Amsterdam, September 15, 2014

Contents

1 Introduction

Information retrieval (IR) is about finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers). Originally, information retrieval was an activity that only a few people engaged in: reference librarians, paralegals, and similar professional searchers (Manning et al., 2008). Now the world has changed, and hundreds of millions of people engage in information retrieval every day when they use a web search engine, search their email, or go online to make a purchase. Information retrieval is fast becoming the dominant form of information access, overtaking traditional database-style searching (Manning et al., 2008). As information retrieval technology has been playing a more and more important role in people's daily lives, a large number of effective information retrieval algorithms has been proposed to assist people to find their underlying information need. For instance, Balog et al. (2006) proposed generative probabilistic models for expert searching given an organization's document repositories.

In many scenarios that employ IR technology, a *query* consists of a set of words that is submitted by users to express their underlying information need, and a *result list* is a ranking of a set of documents that try to respond to the query. Here, a result list can be generated by a wide range of current IR approaches, which can be based on various search models such as Boolean (Joyce and Needham, 1958), Vector Space (Salton and Lesk, 1968), Probabilistic Retrieval (Maron and Kuhns, 1960) models. The result lists produced by these approaches depend on the exact definition of the underlying concept of relevance. Some IR approaches may perform well in a specific setting, that is, retrieve relevant documents in response to a user request for that specific IR applications, but may perform poor if the setting of the applications changes. In the past decades, researchers in IR strive to improve the retrieval performance for a range of different settings and applications, and one way of doing so is to generate multiple alternative result lists so as to generate a better ranking in response to a query. *Data fusion* approaches, also called *rank aggregation* approaches, consist in combining result lists produced by different retrieval algorithms in order to generate a new and hopefully better ranking. Fig. 1.1 illustrates the role of data fusion in IR.

Research carried out in the area of data fusion in the past years has argued that data fusion has the potential of combining effectively all the various sources of evidence considered in various input methods (Fox and Shaw, 1994; He and Wu, 2008; Kozorovitsky and Kurland, 2011; Shaw and Fox, 1993; Sheldon et al., 2011; Tsai et al., 2008). As

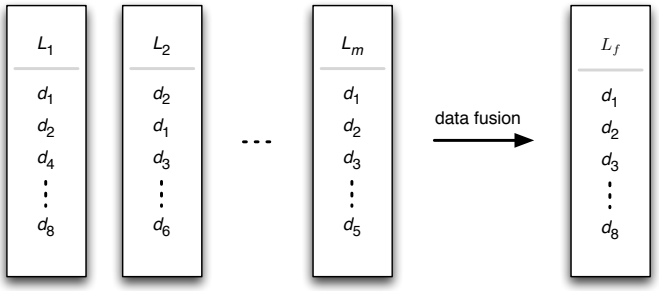


Figure 1.1: Data fusion strategies in IR. The input of a data fusion method is a set of result lists generated by different retrieval methods in response to a query, i.e., L_1, L_2, ..., L_m, and the output is a fused list L_f generated by a data fusion method.

shown by previous work on data fusion, documents are more likely to be relevant if they appear in the result lists of the majority of the retrieval algorithms (Beitzel et al., 2003; Dong and Srivastava, 2013; Montague and Aslam, 2002; Shokouhi and Si, 2011; Wu, 2012). Previous work has also shown that different retrieval algorithms often return very different irrelevant documents, but many of the same relevant documents. Moreover, researchers found that data fusion approaches improve the performance over the individual approaches that they fuse, even when some of these have a weak retrieval performance. Previous data fusion approaches also tend to smooth out biases of the individual retrieval approaches.

In this thesis we continue the research on data fusion. When data fusion is employed in new scenarios, for instance searching posts on microblogging platforms, such as Twitter,[1] new issues arise. On microblogging platforms, people tend to talk about a topic mostly during very specific short time intervals (Zhao and Rosson, 2009). We investigate the principle that data fusion may help to enhance the retrieval performance of searching posts by using the characteristics of microblog environments, and translate our insights into new time-aware data fusion algorithms for effective microblog search. Also, we revisit some basic principles behind data fusion, resulting in a new data fusion algorithm that first constructs *manifolds* (Thurston and Milnor, 1979) for documents based on their inter-document similarities and then lets highly ranked documents boost the fusion scores of documents that are ranked low in the same manifolds.

A second main line of work in the thesis is based on the observation that a query does not always form an unambiguous reflection of a user's information need. Recently, *search result diversification* is widely being studied as a way of tackling query ambiguity (Agrawal et al., 2009). Instead of trying to identify the "correct" interpretation behind an ambiguous query, the idea is to make the search results diversified so that users with different backgrounds and different intents will find at least one of these results to be relevant to their information need. In contrast to the traditional assumption of independent document relevance (Robertson, 1977), search result diversification approaches typically

[1] `http://www.twitter.com`

consider the relevance of a document in light of other retrieved documents. Diversification algorithms try to identify the probable *aspects* of the query and attempt to return documents for each aspect, thereby making the result list more diverse. In this thesis, we examine the hypothesis that data fusion can improve retrieval performance in terms of diversity metrics by promoting aspects that are found in disparate ranked lists to the top of the fused list, which results in a new diversified data fusion method.

Unlike search result diversification, *personalized web search* strives to enhance the retrieval system's knowledge about users' information needs (Vallet et al., 2010). Rather than aiming to satisfy as many users as possible, personalization aims to build a sense of who the user is, and maximize the satisfaction of a specific user (Bennett et al., 2012). Although different, diversification and personalization are not incompatible and do not have mutually exclusive goals (Vallet and Castells, 2012). Indeed, search results generated by diversification techniques should be more diverse when a user's preferences are unknown to the search engine. Likewise, personalization can improve the effectiveness of aspect weighting in diversification, by favoring query interpretations that are predicted to be more related to each specific user. This thesis investigates the problem of personalized diversification of search results, and translates our insights on this research topic into a new approach for enhancing both diversification and personalization performance.

1.1 Research Outline and Questions

The broad question that motivates the research in the thesis is: *How can we improve the performance of data fusion and diversification in information retrieval?* Individual components for solving this problem already exist (see Chapter 2 for an overview), but other aspects, such as the relationship between data fusion and diversification, have not yet been investigated. This thesis aims to close some of these gaps, contributing new data fusion and search result diversification solutions to the field of IR.

We start our investigation by focusing on employing data fusion approaches for searching posts in a microblogging environment, as previous research has found that data fusion can enhance the retrieval performance in many cases (Dong and Srivastava, 2013; Fox and Shaw, 1994; Wu, 2012). Microblogging platforms, such as Twitter, have become indispensable communication channels through which hundreds of millions of users around the world witness breaking news events. The characteristics of the posts, such as their limited length, along with easy access on many platforms, lead to regular status updates by large numbers of people (Zhao and Rosson, 2009). Microblogging platforms display fast paced dynamics as reflected by rapidly evolving topics (Yang and Leskovec, 2011). Searching posts in such rapidly changing environments is a challenge (Lin et al., 2012). To tackle this problem, much previous work has focused on a range of content-based criteria for ranking posts in response to a query, in combination with a broad range of other ranking criteria, including, e.g., existence of hyperlinks, hashtags and retweets.

Data fusion is a popular method for generating result lists based on multiple ranking criteria. This thesis looks at the problem of searching microblog posts as a data fusion task, and answers the following questions:

RQ 1 Can data fusion help microblog search?

In answering this question, we first directly use existing data fusion approaches for searching posts. After analyzing previously developed data fusion approaches in microblog search, we find that the assumptions made in previous fusion approaches do not work well in the microblog environments. In particular, previous work has assumed that only documents ranked high in many of the input result lists can be ranked high in the final fused list (Fox and Shaw, 1994; He and Wu, 2008; Shaw and Fox, 1993; Tsai et al., 2008). To address this shortcoming of existing data fusion approaches in microblog environments, we develop a novel probabilistic data fusion model, BurstFuseX, that is burst-aware and not only utilizes information traditionally used when merging ranked lists, such as ranks of documents, but also exploits temporal information, i.e., the publication timestamps of microblog posts.

In previous data fusion approaches, documents in the fused list are ranked in decreasing order of their fusion scores. The fusion score of a document is usually the sum of rank scores from the individual input lists. Previous work on data fusion often assumes, either implicitly or explicitly, that the rank score of a document is set to zero if the document does not appear in the input list. We challenge this assumption. An extension of BurstFuseX is derived to infer scores of so-called missing documents during aggregation for microblog search. Here, a *missing document* for a result list L is a document that does not appear in the list L but it does appear in other result lists that we aim to fuse. We propose a method based on matrix factorization to infer scores of missing documents and ask the following questions:

RQ 2 How to infer scores of so-called missing documents in data fusion?

We then turn to issues of using data fusion in a generic document search setting. Much of the past work on data fusion assumes that documents in the lists to be fused are independent and that only documents that are ranked high in many of the lists are likely to be relevant in response to a given query (Das-Gupta and Katzer, 1983; Griffiths et al., 1986; Shaw and Fox, 1993; Wu, 2012). As a consequence, a relevant document that is ranked low in a list, and appears only in this single list, will be ranked low in the final fused list. This may be a problem because a low ranked document in a result list does not necessarily mean that it is not relevant. Recent work has become aware of this problem and tries to tackle it based on clustering: documents appearing in the lists to be fused are clustered and the score of a document that appears low in a single list can be boosted if it is similar to other relevant documents in the same cluster (Kozorovitsky and Kurland, 2011). While intuitive, such a fusion strategy can fall short in some cases. For instance, a non-relevant document should not be "promoted" even if it is in a cluster that contains a large number of relevant documents. What is worse, cluster-based data fusion may be a bottleneck in some applications. For instance, it is computationally prohibitive in the situation that a large number of documents need to be fused. We investigate a known cluster-based data fusion method analytically and experimentally, and introduce an alternative based on manifolds. We ask the following questions:

RQ 3 Can manifolds be used to improve data fusion performance for ad hoc search?

As we will see in Chapter 6, manifolds allow us to capture a richer inter-document structure than mere clustering. Next, we turn to the application of data fusion in the area of

search result diversification. As we will see in Chapters 4, 5, and 6, data fusion methods can improve retrieval performance in terms of traditional relevance-oriented metrics like MAP and precision@k over the methods used to generate the individual result lists being fused (Fox and Shaw, 1994; Lee, 1995, 1997; Wu, 2012). One reason is that retrieval approaches often return very different irrelevant documents, but many of the same relevant documents (Wu, 2012). We examine the hypothesis that data fusion can improve performance in terms of diversity metrics by promoting aspects that are found in disparate ranked lists to the top of the fused list. Specifically, we propose a fusion-based diversification method, diversified data fusion, which infers latent topics from ranked lists of documents produced by standard fusion methods, and combines this with a state-of-the-art result diversification model. Based on traditional and novel fusion methods, we provide an answer to the following:

RQ 4 Can data fusion help search result diversification?

In our last step in studying search result diversification, we turn to the problem of *personalized diversification* of search results, with the goal of enhancing the performance of both plain search result diversification and plain personalization algorithms. In both search result diversification and personalized web search, an issued query is often viewed as an incomplete expression of a user's underlying need (Shen et al., 2005). Unlike search result diversification, where the system accepts and adapts its behavior to a situation of uncertainty, personalized web search strives to change this situation by enhancing the system's knowledge about users' information needs. Rather than aiming to satisfy as many users as possible, personalization aims to build a sense of who the user is, and maximize the satisfaction of a specific user (Vallet and Castells, 2012). We propose a personalized diversification method based on structured learning. The research questions addressed by our subsequent research are:

RQ 5 How to enhance both diversification and personalization performance at the same time in a supervised way?

In each of the research chapters (Chapters 4–8) we seek answers to the research questions listed above. The answers are given in the conclusions of each chapter and are summarized in Chapter 9 of this thesis.

In the next sections we list the contributions that this thesis makes to the field and we give an overview of the thesis and of the origins of the material.

1.2 Main Contributions

The main contributions of this thesis are listed below.

- **Burst-aware data fusion algorithm for microblog search** – We frame the microblog post search problem as a data fusion problem. We propose a burst-aware data fusion algorithm, BurstFuseX, that not only utilizes a microblog post's ranking information but also exploits its publication time. The proposed framework, BurstFuseX, builds on an existing fusion method and rewards posts that are ranked

highly in many of the result lists being aggregated. An empirical evaluation of our burst-aware data fusion algorithm is carried out using datasets made available by the TREC (Text REtrieval Conference)[2] microblog tracks. This contribution provides an answer to RQ 1.

- **Time-aware rank aggregation algorithm for microblog search** – We propose a second rank aggregation algorithm for searching microblog posts, called TimeRA, that is both time-aware and able to infer the rank scores of missing documents via latent factor modeling. We examine the relative contributions of the main ingredients of TimeRA, i.e., burst detection and latent factor-based score inference. We provide a detailed analysis of the performance of TimeRA and offer a number of examples where we observe the effect hypothesized in microblogging environment that posts in bursts having their rank score boosted. Together with the previous contribution, this one helps answer RQ 2.
- **Effective and efficient data fusion algorithm for ad hoc search** – We revisit the problem of data fusion for ad hoc search. We propose an efficient manifold-based data fusion algorithm, ManX, that not only utilizes the ranks of documents in the result lists but also produces an additional source of rich inter-document similarity information. We experimentally verify the retrieval effectiveness of this algorithm and also show it runs faster than state-of-the-art cluster-based data fusion algorithms. This contribution provides an answer to RQ 3.
- **Diversified data fusion algorithm for search result diversification** – We adopt a new perspective on the problem of search result diversification, based on data fusion. Starting from the hypothesis that data fusion can improve performance in terms of diversity metrics, we examine the impact of standard data fusion methods on search result diversification. We also introduce a new data fusion algorithm, called diversified data fusion, that infers latent topics of a query using topic modeling. We analyze the effectiveness of existing data fusion and our diversified data fusion algorithms for search result diversification. This contribution helps us to answer the research question asked by RQ 4.
- **Structured learning algorithm for personalized diversification** – To further enhance the performance of personalized search result diversification, we set up a structured learning framework for conducting supervised personalized diversification, in which we add effective features. We define two additional constraints in our structured learning framework to ensure that search results are both diversified and consistent with a user's interest. We conduct experiments on an open personalized diversification dataset to analyze the effectiveness of our proposed personalized diversification learning algorithm. This is our final contribution of the thesis, and it helps answer the research question asked by RQ 5.

1.3 Thesis Overview

Besides the current chapter, the thesis consists of two chapters covering the experimental prerequisites and methodology, five research chapters containing our core contributions

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

plus a concluding chapter:

Chapter 2 – Background: Here, we present the background for all subsequent chapters. We mainly focus on data fusion and search result diversification in Information Retrieval.

Chapter 3 – Experimental methodology: We provide details on experimental settings that recur in various chapters of this thesis. Amongst others, we discuss document collections, topic sets, and evaluation metrics. We provide details on our baseline retrieval models, i.e., data fusion and search result diversification models for IR, which recur in Chapters 4–8.

Chapter 4 – Burst-aware data fusion form microblog search: We consider the problem of searching posts in microblog environments. We frame this microblog post search problem as a data fusion problem. We propose BurstFuseX, a fusion model that not only utilizes a microblog post's ranking information but also exploits its publication time. We experimentally verify the effectiveness of the proposed late data fusion algorithm, and demonstrate that in terms of mean average precision it significantly outperforms the standard, state-of-the-art fusion approaches as well as burst or time-sensitive retrieval methods.

Chapter 5 – Time-aware data fusion for microblog search: Similar to Chapter 4, we frame the problem of microblog search as a data fusion problem, but in a different way. We propose a rank aggregation method, TimeRA, that is able to infer the rank scores of documents via latent factor modeling. It is time-aware and rewards posts that are published in or near a burst of posts that are ranked highly in many of the lists being aggregated. Our experimental results show that it significantly outperforms state-of-the-art data fusion and time-sensitive microblog search algorithms.

Chapter 6 – Manifold-based data fusion of ranked lists: We address the task of merging ranked lists that are retrieved in response to a query in an ad hoc search setting. We propose an efficient manifold-based data fusion approach, ManX, that not only utilizes the ranks of documents in the lists but also an additional source of rich inter-document similarity information. We experimentally verify the retrieval effectiveness of our ManX algorithm, and demonstrate that it significantly outperforms the standard fusion approaches that it integrates as well as state-of-the-art cluster-based methods and runs faster than cluster-based methods.

Chapter 7 – Fusion helps diversification: Starting from the hypothesis that data fusion can improve retrieval performance in terms of diversity metrics, we examine the impact of standard data fusion methods on search result diversification. We also introduce a new data fusion method, called diversified data fusion, which infers latent topics of a query using topic modeling, without leveraging outside information. Our experiments show that data fusion methods can enhance the performance of state-of-the-art diversification algorithms and our diversified data fusion significantly outperforms existing data fusion methods in terms of diversity metrics.

Chapter 8 – Personalized search result diversification via structured learning: We set up a structured learning framework for conducting supervised personalized diversification, in which we add features extracted directly from documents and

existing diversification algorithms, and, importantly, those generated from topic models. We also define two constraints to ensure that search results are both diversified and consistent with a user's interest. We conduct experiments on an open personalized diversification dataset and find that our supervised learning strategy outperforms unsupervised personalized diversification methods as well as other plain personalization and plain diversification methods.

Chapter 9 – Conclusions: We go back to the research questions introduced in this chapter and provide answers. Finally, we discuss future directions of research.

Chapter 2 serves as background to the research in the technical chapters and can be read if additional insight in the field is required. Chapter 3 provides necessary information on the test collections and evaluation metrics that are used in the technical chapters and gives additional details on the baseline data fusion and search result diversification models. Each of the research chapters, Chapters 4 to 8, can be read individually, as the contents of these chapters is not dependent on other research chapters. Finally, reading only this introduction chapter and the conclusion in Chapter 9 gives a dense summary of the whole thesis, and provides answers to the research questions.

1.4 Origins

The following publications form the basis of chapters in this thesis:

- Chapter 4 is based on Liang and de Rijke (To appear): Burst-aware data fusion for microblog search, *Information Processing & Management*, To appear.
- Chapter 5 is based on Liang et al. (2014d): Time-aware rank aggregation for microblog search. CIKM 2014.
- Chapter 6 is based on Liang et al. (Submitted): Efficient Manifold-based fusion of ranked lists, Submitted to a journal.
- Chapter 7 is based on Liang et al. (2014a): Fusion helps diversification, SIGIR 2014.
- Chapter 8 is based on Liang et al. (2014b): Personalized search result diversification via structured learning, KDD 2014.

Finally, this thesis draws from insights and experiences gained in:

- Cai et al. (2014b): Time-sensitive personalized query auto-completion, CIKM 2014.
- Cai et al. (2014a): Personalized document re-ranking based on bayesian probabilistic matrix factorization, SIGIR 2014.
- Liang and de Rijke (Submitted): Formal language models for finding groups of experts, Submitted to a journal.

- Liang et al. (2014c): The impact of semantic document expansion on cluster-based fusion for microblog search, ECIR 2014.
- Liang and de Rijke (2013): Finding knowledgeable groups in enterprise corpora, SIGIR 2013.
- Liang et al. (2013): Late data fusion for microblog search, ECIR 2013.
- Ren et al. (2014): Hierarchical multi-label classification of social text streams, SIGIR 2014.
- Ren et al. (2013): Personalized time-aware tweets summarization, SIGIR 2013.

2 Background

In this chapter, we provide background material for later chapters in this thesis. We start with an introduction to information retrieval in Section 2.1, where we discuss main retrieval models in IR. In Section 2.2 we detail the ad hoc, microblog search and search result diversification tasks that we will deal with in the following research chapters, i.e., Chapters 4–8. In 2.3 we provide background methods that will be used to deal with our tasks. Specifically, in 2.3.1 we detail data fusion methods. Because our proposed data fusion algorithms rely on some lists produced by microblog retrieval algorithms, we briefly describe related background on microblog retrieval in Section 2.3.2. Our proposed data fusion algorithms utilize latent factor modeling and manifold-based algorithms, and methods that are briefly described in Section 2.3.3 and Section 2.3.4, respectively. We give a brief background of previous methods on search result diversification as well as personalized diversification in Section 2.3.5. Our proposed diversified data fusion and personalized diversification algorithms proposed in Chapter 7 and Chapter 8, respectively, work with latent topic modeling; thus we also briefly recall methods for latent topic modeling in Section 2.3.6. Finally, we detail background of structured learning methods in Section 2.3.7 for our proposed personalized diversification algorithm.

2.1 Information Retrieval

Information retrieval (IR) is about finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers) (Manning et al., 2008). IR deals with the representation, storage, organization of, and access to information items (Baeza-Yates and Ribeiro-Neto, 2011). After the term IR had been coined in 1950 (Mooers, 1960), a large number of retrieval algorithms have been proposed. These retrieval algorithms can be simply categorized into at least the following retrieval models: Boolean retrieval models (Joyce and Needham, 1958), vector space models (Salton and Lesk, 1968), probabilistic retrieval models (Maron and Kuhns, 1960), language retrieval models (Ponte and Croft, 1998) and learning to rank retrieval models (Liu, 2009).

Among the models just listed, Boolean retrieval models are the simplest retrieval ones, where the input query is in the form of a Boolean expression of terms, and combined with operators, such as AND, OR, and NOT. The operator NOT indicates, for instance, that the terms should not be included in relevant documents. Documents to be

returned in response to the query are also viewed as a set of terms. The decision about the relevancy of a document is a binary one, a document is either relevant and therefore included in the set of retrieved documents, or it is not relevant and is thus ignored. The advantages of Boolean retrieval models are obvious. Terms in the query all need to be exactly matched, which usually results in returning too few or too many documents. For the retrieval systems, it is difficult to rank the output documents as the documents are considered as relevant or not only. Sometimes for a user, it is hard to translate his information need into an input query with Boolean expression (Chang et al., 1999). All terms are considered to have the same weights in the retrieval systems.

Because of many disadvantages in Boolean retrieval models, researchers in IR then proposed the second generation models, which is *vector space models* (Salton and Lesk, 1968). In these retrieval models, documents and queries are represented as "bag-of-words." Each term in a candidate document has its own weight, and documents are ranked by decreasing cosine value between documents and the query. The well-known tf-idf weighting scheme plays an important role in vector space models. The weighting scheme makes these two assumptions: (i) a term that appears in many documents should not be regarded as more important than one that appears in few documents; (ii) a document with many occurrences of a term should not be regarded as less important than a document with few occurrences of the term. The advantages of vector space models are that documents can be ranked in the output set, terms are weighted by their importance and the matching between document and a input query can be partial. In contrast, the disadvantages of the models are that terms are assumed to be independent from each other and weighting of terms or documents is intuitive but not quite formal.

One of the most effective retrieval models in many IR applications are *probabilistic models*. The probabilistic approach to retrieval was first presented in (Maron and Kuhns, 1960), and the Probability Ranking Principle (PRP) (Maron and Kuhns, 1960) that represents the theoretical justification of probabilistic IR models was proposed in (Robertson and Jones, 1976). In these models the probability of a document being relevant to a users underlying information need presented by a query needs to be estimated. As the initial probabilistic model explicitly contains the probability of a document being relevant and the probability of the same document not being relevant (Robertson, 1977; Robertson and Jones, 1976), it is often referred to as independence retrieval model. The success of this retrieval model depends on the availability of the distributions of terms over relevant and non-relevant documents and these distributions are usually unknown. At the very beginning of the development of probabilistic models, the initial models use binary weights for query terms in documents; later they were changed to include term frequencies (Robertson et al., 1980).

Instead of directly computing the probability of a document relevant to a given query as in probabilistic models, *language models* tackle the problem of retrieving documents in a different way. Inspired by the fact that users tend to think of words that may appear in a relevant document when they try to come up with words for the input query, language models are directly motivated by this idea. A document is a good match to a query if the document model is likely to generate the query, which will in turn happen if the document contains the query words often. Well-known existing language models are Jelinek-Mercer language model and Dirichlet language model (Ponte and Croft, 1998). In recent years, with the development of technology in machine learning, many

supervised learning algorithms have been proposed for retrieving documents, which can be called *learning to rank* models (Liu, 2009). Many existing ranking models contain parameters. In order to get a reasonably good ranking performance, learning to rank algorithms need to tune these parameters using a validation set (Liu, 2009). Often assumptions are made in learning to rank models. For instance, it is supposed that there is labelled data for training a retrieval model.However, in some cases the labelled data are not available as manually labeling data is expensive.

In this thesis, we do not try to propose a new retrieval model but instead we try to combine the search result lists generated by these retrieval models into a final ranked list, so that the performance of the fused list is better than that of the individual lists. This is the so-called *data fusion* problem, for which we will give background in the Section 2.3. Befor continuing our literature review with work related to data fusion and search result diversification, we first describe the retrieval tasks that we will address.

2.2 Tasks

To answer the research question listed in Chapter 1, we need to address three main tasks in this thesis: (i) ad hoc search; (ii) microblog search; (iii) search result diversification. We briefly describe the tasks in the following sections.

2.2.1 Ad hoc search

Given a query $q = \langle q_1, q_2, \ldots, q_{|q|} \rangle$ that captures a user's underlying information need and a set of documents $\mathcal{C} = \{d_1, d_2, \ldots, d_{|\mathcal{C}|}\}$, the task of ad hoc search is to define a ranking function $f(d|q)$ that assigns a rank score to each document $d \in \mathcal{C}$ with regard to the input query q and then the retrieval model will rank the documents with decreasing scores of the documents. Here q_i is a token in query q, d_i is a document in $\mathcal{C}$, $|q|$ is the total number of tokens in q and $|\mathcal{C}|$ is the total number of documents in $\mathcal{C}$. The definition of the ad hoc search task in IR can be formulated as:

$$q = \langle q_1, q_2, \ldots, q_{|q|} \rangle, \mathcal{C} \xrightarrow{f(d|q)} L,$$

where L is a result list generated by the ad hoc retrieval system. We will give our solution for the ad hoc search task in Chapter 6.

2.2.2 Microblog search

In TREC, the microblog tracks (Clarke and Craswell, 2011; Soboroff et al., 2012) addressed a single pilot task, entitled microblog search task, where the user wishes to see the most recent yet relevant information to the input query. The microblog search task can be summarized as: at time t, find tweets about an input query q. This task is akin to ad hoc search on Twitter, where a user's information need is represented by a query at a specific time. Participants were asked to rank the relevant tweets by time (Clarke and Craswell, 2011). The definition of the microblog search task can be formulated as:

$$q = \langle q_1, q_2, \ldots, q_{|q|} \rangle, t, \mathcal{C} \xrightarrow{f(d|q)} L,$$

where $\mathcal{C}$ is a set of documents, i.e., a set of posts in the microblog search task, and d is a document, i.e., a post in the collection. One possible interpretation of the task is to rank all tweets up to time t by ad hoc search algorithm, and then discard irrelevant tweets. We will discuss our solutions for microblog search task in Chapters 4 and 5.

2.2.3 Search result diversification

In both ad hoc search and microblog search tasks, the search algorithms mainly focus on the relevance of the documents and ignore the novelty of the returned documents. In recent years, the task of search result diversification has gained much attention, and many algorithms have been proposed to tackle the challenges within the task. The search result diversification task is akin to ad hoc search, but differs in that one tries to address as many different intents as possible. Here we formally give the definition of the task. Let $\mathcal{C}_q$ denote the ranking produced in response to the input query q by a relevance-oriented ranking approach. In addition, let $\mathcal{N}_q$ and $\mathcal{N}_d$ denote the sets of intents (aspects) for which the query q and each document $d \in \mathcal{C}_q$ are relevant, respectively. The task of search result diversification is to find a result list $L \in 2^{\mathcal{C}_q}$ such that the documents in L cover as many intents (aspects) as possible. The definition of the search result diversification task can be formulated as:

$$q = \langle q_1, q_2, \ldots, q_{|q|} \rangle, \mathcal{C} \overset{f(d|q)}{\longrightarrow} L = \arg\max_{L' \in 2^{\mathcal{C}_q}} \left| \bigcup_{d \in L'} \mathcal{N}_q \cap \mathcal{N}_d \right|, \text{s.t.} |L'| \leq k,$$

where $k > 0$ is the diversification cutoff, denoting the number of top documents from $\mathcal{C}_q$ to be diversified, and $2^{\mathcal{C}_q}$ is the power set of $\mathcal{C}_q$.The subset with the maximum number of covered aspects of the query up to the cutoff k is chosen as the optimal diversified result ranking L.

2.3 Methods

The previous section has introduced the tasks that we address in this thesis. In this section, we provide overviews of previous methods tackling these three tasks. As we utilize data fusion and latent factor modeling approaches to tackle the task of microblog search in Chapters 4 and 5, we provide a brief overview of data fusion (in Section 2.3.1), microblog search (in Section 2.3.2) and latent factor modeling (in Section 2.3.3) methods. As we utilize manifold-based and data fusion approaches to tackle the task of ad hoc search in Chapter 6, we provide a background overview of manifold-based methods (in Section 2.3.4). Finally, as we also utilize topic modeling and structured learning approaches to tackle the task of search result diversification in Chapters 7 and 8, respectively, we provide an overview of search result diversification (in Section 2.3.5), topic modeling (in Section 2.3.6), and structured learning (in Section 2.3.7) methods.

Before we move on to the methods, we summarize the main notation used in the thesis in Table 2.1.

Table 2.1: Main notation used in the thesis.

Notation	Gloss
$\mathcal{C}$	corpus of microblog posts
q	query
d	document
L	ranked list of documents
$\|L_i\|$	length of list L_i
$\mathbf{L}$	set of ranked lists
$\mathcal{C}_\mathbf{L}$	a set of documents appearing in component lists to be fused
$\|\mathcal{C}_\mathbf{L}\|$	number of documents in $\mathcal{C}_\mathbf{L}$
X	standard fusion method
$f_X(d;q)$	score of post d for query q according to standard fusion method X
$R_{L_i}(d)$	rank-based score of d in list L_i
$\text{rank}(d, L_i)$	rank of d in list L_i
$\mathbf{R}$	$m \times \|\mathcal{C}_\mathbf{L}\|$ list-post rank score matrix
$\mathbf{S}$	matrix used for inferring latent topics for $\mathbf{L}$
$\mathbf{V}$	matrix used for inferring latent topics for $\mathcal{C}_\mathbf{L}$
$\mathbf{S}_i$	a column vector in $\mathbf{S}$ used for aspects of L_i
$\mathbf{V}_j$	a column vector in $\mathbf{V}$ used for aspects of d_j
t_i	timestamp
d_{t_i}	document with timestamp t_i
b	a burst of documents
$\mathcal{B}$	set of all bursts in $\mathcal{C}_\mathbf{L}$
I_{ij}	indicate whether $d_j \in L_i$
$\mathbf{W}$	weight matrix for $d \in \mathcal{C}_\mathbf{L}$
$\mathbf{D}$	diagonal matrix produced by $\mathbf{W}$
$\mathbf{I}$	identity matrix
w	a token
N_d	number of tokens in d
m	number of ranked list to be fused, i.e., $m = \|\mathbf{L}\|$
T	number of topics
V	number of unique tokens in $\mathcal{C}_\mathbf{L}$
θ_d	multinomial distribution of topics specific to d
z	topic denoted in latent topic literature
ϕ_z	multinomial distribution of tokens specific to topic z

2.3.1 Data fusion

The task of fusing document lists that have been retrieved in response to a given query so as to compile a single more effective result list has been widely studied in the information retrieval literature (Beitzel et al., 2003; Dong and Srivastava, 2013; Fox and Shaw, 1994; He and Wu, 2008; Kozorovitsky and Kurland, 2011; Montague and Aslam, 2002; Shaw and Fox, 1993; Sheldon et al., 2011; Shokouhi and Si, 2011; Tsai et al., 2008; Wu, 2012). Data fusion has a long history with the CombSUM family of fusion methods being the oldest and one of the most successful ones in many IR tasks (Liang et al., 2013; Shaw and Fox, 1993). The lists are often produced by multiple ranking functions, e.g., query representations or document representations (Croft, 2000). Data fusion has a large

number of applications, e.g., in multilingual information retrieval (Sheldon et al., 2011; Si et al., 2008), federated search (He et al., 2011; Hong and Si, 2012; Shokouhi and Si, 2011) also known as distributed retrieval (Crestani and Markov, 2013), resource selection (Hong and Si, 2013; Markov and Crestani, 2014; Markov et al., 2013b), etc. We divide the existing data fusion approaches into supervised and unsupervised methods.

Supervised data fusion approaches first extract a large number of features, either from documents or lists, and then utilize a machine learning algorithm to train the fusion model (Croft, 2000; Efron, 2011; Sheldon et al., 2011; Tsai et al., 2008; Wu, 2012). Liu et al. (2007) set up a general framework for conducting supervised data fusion, in which learning is formalized as an optimization problem in which one minimizes disagreements between ranking results and the labeled data. Tsai et al. (2008) propose a learning approach for the merging process in multilingual information retrieval. For their learning data fusion approach, the authors extract a number of features from the given query, the documents to be retrieved and the translation, and then use an existing learning to rank algorithm to construct a merge model from a large amount of labeled data. Qin et al. (2010) propose a supervised probabilistic data fusion model, which is based on coset-permutation distance and defined in a stage-wise manner. To fuse result lists generated by different query reformulations, the state-of-the-art data fusion method λ-Merge proposed by Sheldon et al. (2011) first extracts features from both the lists and the documents appearing in the lists, and then uses a learning to rank method to optimize a given metric, like NDCG, MAP, to fuse the lists into a final merged list in response to a given query. We use λ-Merge as a representative example of supervised fusion methods. Recently, Hong and Si (2012) propose a novel supervised fusion model for result merging by utilizing multiple centralized retrieval algorithms. However, the fact that a large amount of labeled data has to be available, together with other supervised problems (for instance over-fitting noted above in λ-Merge), makes supervised data fusion less useful when labeled data is hard to come by. Our experimental results in Chapters 4 and 5 show that in many cases, even traditional unsupervised data fusion methods are able to beat state-of-the-art supervised data fusion methods.

In contrast, *unsupervised* data fusion methods mainly utilize either retrieval scores or ranks of documents in the lists to be merged (Bruno and Marchand-Maillet, 2009; Croft, 2000; Fox and Shaw, 1994; Khalaman and Kurland, 2012; Shaw and Fox, 1993; Wu, 2012). Methods utilizing retrieval scores take score information from the result lists to be fused as input, while those utilizing rank information only use order information of the documents appearing in any of the lists to be fused as input. Data fusion methods utilizing rank information have many uses and applications in information retrieval, including, e.g., meta-search (Aslam and Montague, 2001; Fox and Shaw, 1994) where only order information from the result lists tends to be available. Our data fusion algorithms proposed in Chapters 4 and 5 only use rank information of the posts in the result lists, which makes it usable in cases where only order information is available.

Unsupervised data fusion has a long history with the CombSUM family of fusion methods being the oldest and one of the most successful ones in many information retrieval tasks (Croft, 2000; Kozorovitsky and Kurland, 2011; Lee, 1995; Shaw and Fox, 1993; Tsagkias et al., 2011). Other unsupervised data fusion approaches include, for instance, Borda Count (Aslam and Montague, 2001; Dwork et al., 2001; van Erp and Schomaker, 2000), median data fusion (Fagin et al., 2003), genetic algorithm (Beg,

2004), fuzzy logic-based data fusion (Ahmad and Beg, 2002), Markov Chain-based data fusion (Dwork et al., 2001), the outranking model for fusion (Farah and Vanderpooten, 2007) and a distance-based model (Klementiev et al., 2008). In addition, Markov and Crestani (2014); Markov et al. (2012, 2013a) provide theoretical arguments on why some traditional unsupervised fusion methods work, and based on the insights gained, they propose other unsupervised fusion methods.

Khalaman and Kurland (2012) utilize the content of documents appearing in the result lists to be fused to get an additional source of rich information, i.e., document similarities, and then integrate information induced from the clusters of similar documents created across the result lists to be merged with the output of a fusion method that relies on retrieval scores. This fusion model makes strong assumptions: the content of documents is assumed to be available and it is easy to compute document similarities and get clusters for documents. However, in the case of microblog retrieval, which we consider in Chapters 4 and 5, some of these assumptions are somewhat unrealistic. For instance, some posts with only links but without any words are still labeled as relevant in response to the query, and creating clusters of similar posts may be very challenging as the length of posts is at most 140 characters, while many posts are ambiguous (Zhao and Rosson, 2009). In addition, many effective fusion methods are based on the assumption that only documents that are highly ranked in many of the lists are likely to be relevant (Aslam and Montague, 2001; Croft, 2000; Dwork et al., 2001; Fox and Shaw, 1994; Kozorovitsky and Kurland, 2011; Lee, 1995; Montague and Aslam, 2002; Tsagkias et al., 2011). As a consequence, a relevant document will be ranked low in the final fused list if it appears only in a single list and is ranked low in this list.

In Chapters 4–6, three data fusion baselines, CombSUM, CombMNZ and Cluster-based data fusion methods, are frequently used to see the improvements of our proposed data fusion algorithms. Thus, we detail these data fusion baseline methods in this subsection.

CombSUM and CombMNZ. Let R_{id} denote the score of document d based on the rank of d in list L_i; in the literature on data fusion, one often finds $R_{id} = 0$ if $d \notin L_i$ (d still in the combined set of documents $\mathcal{C}_{\mathbf{L}} := \bigcup_{i=1}^{m} L_i$). In both CombSUM and CombMNZ, R_{id} is often defined as:

$$R_{id} = \begin{cases} \frac{(1+|L_i|)-\mathrm{rank}(d,L_i)}{|L_i|} & d \in L_i \\ 0 & d \notin L_i, \end{cases} \tag{2.1}$$

where $|L_i|$ is the length of L_i and $\mathrm{rank}(d, L_i) \in \{1, \ldots, |L_i|\}$ is the rank of d in L_i. The well-known CombSUM fusion method (Fox and Shaw, 1994; Wu, 2012), for instance, scores d by the sum of its rank scores in the lists:

$$f_{\mathrm{CombSUM}}(d; q) := \textstyle\sum_{L_i} R_{id},$$

while CombMNZ (Fox and Shaw, 1994; Wu, 2012) rewards a document d that ranks high in many lists:

$$f_{\mathrm{CombMNZ}}(d; q) := |\{L_i : d \in L_i\}| \cdot f_{\mathrm{CombSUM}}(d; q),$$

where $|\{L_i : d \in L_i\}|$ is the number of lists in which d appears.

Cluster-based data fusion. To improve performance of unsupervised data fusion approaches, Kozorovitsky and Kurland (2011) considered the *cluster hypothesis* and proposed a cluster-based fusion method, ClustFuse, where documents in the same cluster provide relevance-status support to each other. Their ClustFuse algorithm can be formulated as follows:

$$f_{\text{ClustFuse}}(d;q) := (1-\lambda)p(d|q) + \lambda \sum_{c \in Cl(\mathcal{C}_{\mathbf{L}})} p(c|q)p(d|c), \tag{2.2}$$

where $p(d|q)$ is the probability that a document d is relevant to a query q, $Cl(\mathcal{C}_{\mathbf{L}})$ is a set of clusters generated from documents in $\mathcal{C}_{\mathbf{L}}$, $p(c|q)$ is the probability that a cluster c is relevant to q, $p(d|c)$ is the probability that a document d is associated with a cluster c and λ is a parameter.

Three key components need to be estimated in (2.2): $p(d|q)$, $p(c|q)$ and $p(d|c)$. Using Bayes' rule, $p(d|q)$ can be rewritten as $p(d|q) = \frac{p(q|d)p(d)}{p(q)}$. The probability of a query $p(q)$ can be represented as $p(q) = \sum_{d' \in \mathcal{C}_{\mathbf{L}}} p(q|d')p(d')$. The prior distribution of documents in $\mathcal{C}_{\mathbf{L}}$ is assumed to be uniform, i.e., $p(d)$ is a constant. Then, $p(d|q)$ can be rewritten as $p(d|q) = \frac{p(q|d)}{\sum_{d' \in \mathcal{C}_{\mathbf{L}}} p(q|d')}$. If $p(q|d)$ is considered to be proportional to a fusion score $f_{\text{X}}(d;q)$, then $p(d|q)$ can be finally estimated as follows:

$$p(d|q) := \frac{f_{\text{X}}(d;q)}{\sum_{d' \in \mathcal{C}_{\mathbf{L}}} f_{\text{X}}(d';q)}. \tag{2.3}$$

Similarly, assuming a uniform prior for clusters, $p(c|q)$ can be rewritten as $p(c|q) = \frac{p(q|c)}{\sum_{c' \in Cl(\mathcal{C}_{\mathbf{L}})} p(q|c')}$. Here, $p(q|c)$ can be approximated by a product-based representation, i.e., $p(q|c) = \prod_{d \in c} f_{\text{X}}(d;q)$. So the final estimation of $p(c|q)$ is the following:

$$p(c|q) := \frac{\prod_{d \in c} f_{\text{X}}(d;q)}{\sum_{c' \in Cl(\mathcal{C}_{\mathbf{L}})} \prod_{d' \in c'} f_{\text{X}}(d';q)}. \tag{2.4}$$

Assuming a uniform prior for documents in $\mathcal{C}_{\mathbf{L}}$, the last component in (2.2), i.e., $p(d|c)$, can be represented as $p(d|c) = \frac{p(c|d)}{\sum_{d'' \in \mathcal{C}_{\mathbf{L}}} p(c|d'')}$. ClustFuse assumes $p(c|d)$ to be proportional to $\frac{1}{|c|} \sum_{d' \in c} \text{sim}(d', d)$, where $|c|$ is the number of documents in a cluster c, which is chosen to be constant for all clusters: $|c| = \delta$. Also, $\text{sim}(d', d)$ is the similarity score between d' and d. So $p(d|c)$ can be represented as follows:

$$p(d|c) := \frac{\sum_{d' \in c} \text{sim}(d', d)}{\sum_{d'' \in \mathcal{C}_{\mathbf{L}}} \sum_{d' \in c} \text{sim}(d', d'')}, \tag{2.5}$$

Here, $\text{sim}(d_1, d_2)$ is defined as:

$$\text{sim}(d_1, d_2) = \exp\left\{ -\frac{1}{2} \left(\text{KL}(\theta_{d_1} || \theta_{d_2}) + \text{KL}(\theta_{d_2} || \theta_{d_1}) \right) \right\}, \tag{2.6}$$

where θ_d is a language model of a document d, and $\text{KL}(\theta_{d_1} || \theta_{d_2})$ the Kullback-Leibler divergence between θ_{d_1} and θ_{d_2}.

The following sections will briefly review relevant literature with work related to our specific proposed data fusion and search result diversification methods in Chapters 4, 5, 6, 7 and 8.

2.3.2 Microblog retrieval

Microblog retrieval has become an active research topic in IR, especially following the launch of the Microblog track at TREC in 2011 (Lin et al., 2012). Earlier work, however, already explored the task of retrieving microblog posts. O'Connor et al. (2010) present *TweetMotif*, an exploratory search application for Twitter. Unlike traditional approaches to information retrieval, which present a simple list of messages, *TweetMotif* groups messages by frequent significant terms, a result set's subtopics, which facilitate navigation and drilldown through a faceted microblog search interface. Efron (2010) proposes a language model for hashtag retrieval in a microblog environment, where retrieved hashtags on a topic of interest for query expansion are utilized to improve the performance of microblog search. Duan et al. (2010) show that learning to rank methods work well on the task of microblog retrieval and that account authority and URL presence are very strong features.

Following the launch of the Microblog track at TREC in 2011, many approaches have been proposed. Some (Amati et al., 2011; Cao et al., 2011; Horn et al., 2011; Metzler and Cai, 2011; Wei et al., 2011) exploit the idea that microblog queries are distinguished from web queries with many unique characteristics, and utilize the temporal information to help searching posts. The method proposed by Metzler and Cai (2011) combines a Markov random field model with a learning to rank model for searching posts, which achieved the best p@30 performance at TREC in 2011. A combination strategy is also used by Zhang et al. (2011) to search posts, where they combine a field-based model that takes the frequency of a query term in different document fields into account with query expansion. In contrast, work present in (Bandyopadhyay et al., 2011) uses query expansion only for searching posts, but the way their query expansion method works is different; they use the Google Search API to retrieve pages from the web, and use the titles to expand the queries.

At the TREC 2012 Microblog track, Luo et al. (2012) consider a microblog post to be a structured document, consisting not only of the text, but also of other blocks, like hashtags, links, and mentions. Using these blocks as features in a learning to rank method, they show good retrieval performance. At TREC 2012, the best performing run also uses a learning to rank model (Han et al., 2012). Wei et al. (2012) propose a ranking algorithm with temporal information based on a language model. Kim et al. (2012) present two approaches to address the problem of the limited vocabulary of each post due to their short length. The first is query expansion through pseudo-relevance feedback and the other is document expansion of tweets using web documents linked from the body of a tweet. Jabeur et al. (2012) experiment with a Bayesian network retrieval model for post search and a feature learning model for relevance classification.

Beside the approaches presented at TREC 2011 and 2012, many microblog post retrieval approaches have been proposed outside TREC since the launch of the TREC 2011 Microblog track. For instance, Massoudi et al. (2011); Miyanishi et al. (2013a) propose a method for query expansion in the microblog domain and find that this is highly effective. Naveed et al. (2011) explore the impact of document length normalization on retrieval performance and find that this has a negative effect. They also introduce *interestingness* as a measure for microblog posts and show that using this measure leads to better retrieval effectiveness. Choi et al. (2012) suggest a quality model using surrogate

judgments based on retweets that can be collected automatically to train a microblog search model. Chang et al. (2013) propose a method to utilize Twitter TinyURLs (shortened URL links) to detect fresh and high-quality documents, and leverage Twitter data to generate novel and effective features for ranking documents. The work by Miyanishi et al. (2013b), Dakka et al. (2012), Choi and Croft (2012) and Massoudi et al. (2011) utilizes burst (time) information to boost the performance of searching posts, where the bursts are detected based on the terms in the documents.

2.3.3 Latent factor modeling

Latent factor models are often used in collaborative filtering (CF) (Goldberg et al., 1992) and recommender systems (Kurucz et al., 2007; Salakhutdinov and Mnih, 2008a,b). Matrix factorization methods form a group of well-known latent factor models, that include, for instance, singular value decomposition (SVD) (Kurucz et al., 2007; Salakhutdinov and Mnih, 2008b), probabilistic matrix factorization (Salakhutdinov and Mnih, 2008b), social regularization (Ma et al., 2011a) and bayesian approaches (Salakhutdinov and Mnih, 2008a). These methods first model users with latent interests and the products with latent features by matrix factorization, and then try to predict the rating of products for the given users with the observations of the existing users' rating data (Kurucz et al., 2007; Ma et al., 2011a; Salakhutdinov and Mnih, 2008a,b).

To help better understand the proposed algorithms in Chapter 5, we provide details of singular value decomposition (Kurucz et al., 2007; Salakhutdinov and Mnih, 2008b) which is one of the most popular latent factor modeling algorithms in collaborative filtering (Ma et al., 2011a). The SVD method is utilized to approximate a rating matrix $\mathbf{R}$ by minimizing:

$$\frac{1}{2}||\mathbf{R} - \mathbf{U}^\top \mathbf{V}||_F^2, \tag{2.7}$$

where $\mathbf{R} \in \mathbb{R}^{m \times n}$ is an $m \times n$ matrix describing m users' numerical ratings on n items, $\mathbf{U} \in \mathbb{R}^{l \times m}$ is the factorized user-specific matrix, $V \in \mathbb{R}^{l \times n}$ is item-specific matrix and $||\cdot||_F^2$ denotes the Frobenius norm. Both $\mathbf{U}$ and $\mathbf{V}$ serve for making further missing rating data prediction. In collaborative filtering, the matrix $\mathbf{R}$ contains a large number of missing rating data. Hence, we only need to factorize the observed ratings in $\mathbf{R}$, such that (2.7) can be changed to:

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} (\mathbf{R}_{ij} - \mathbf{U}_i^\top \mathbf{V}_j)^2, \tag{2.8}$$

where I_{ij} is the indicator function and $I_{ij} = 1$ if the rating for item j from user i can be observed in $\mathbf{R}$, otherwise $I_{ij} = 0$. Two regularization terms are added into (2.8) to avoid overfitting, such that (2.8) can be changed to:

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} (\mathbf{R}_{ij} - \mathbf{U}_i^\top \mathbf{V}_j)^2 + \frac{\lambda_1}{2}||\mathbf{U}||_F^2 + \frac{\lambda_2}{2}||\mathbf{V}||_F^2, \tag{2.9}$$

where $\lambda_1, \lambda_2 > 0$. Gradient-based approaches can be utilized to get the optimized $\mathbf{U}$ and $\mathbf{V}$ in (2.9). Salakhutdinov and Mnih (2008b) provided a nice probabilistic interpretation with Gaussian observation noise in (2.9).

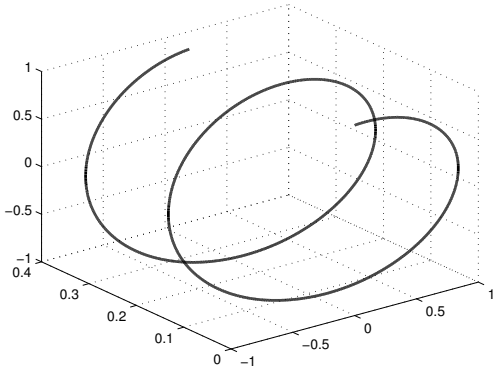


Figure 2.1: A one-dimensional manifold embedded in three dimensions.

2.3.4 Manifold-based algorithms

A *Manifold* is a topological space that resembles Euclidean space near each point (Thurston and Milnor, 1979). Here, we give an example in Fig 2.1 to illustration the concept. Consider the curve which is in $\mathbb{R}^3$. In the figure, the number of dimensions of the curve is somewhat misleading since it seems that the curve can be represented with less than three dimensions. One way of formalizing this intuition is via the idea of *manifold*: the curve is a manifold because it locally "looks like" a copy of less than three dimensions. Many manifold-based algorithms have been proposed to deal with problems in different applications. For instance, in the work of (Griffiths et al., 2012; Zhou et al., 2004; Zhu et al., 2003), they used manifold-based algorithms to address classification tasks of recognizing handwritten digits. In (Griffiths et al., 2012; Wang et al., 2008), manifold-based algorithms were used to tackle the problem of face recognition in an image set, while in (Griffiths et al., 2012) a manifold-based algorithm is used to process a speech signal. One of the most interesting previous publication to us is that in (Diaz, 2005). Based on manifolds, this work assumes that closely related documents tend to be relevant to the same query, and exploits the cluster hypothesis directly by adjusting ad hoc retrieval scores from an initial retrieval so that topically related documents receive similar scores. The previous work in (Diaz, 2005) utilizing query-specific manifolds has focused on reranking a single retrieved list using information induced from manifolds of documents within the list. Manifold-based algorithms were shown to outperforme cluster-based ones. Usually, this is the case because a more complex distance function is needed for the problem at hand than afforded by clustering methods.

To help better understand our proposed method in Chapter 6, here we briefly illustrate how manifold-based algorithm works in IR. Let $\{d_1, \ldots, d_l, d_{l+1}, \ldots, d_n\}$ be given, where the relevance values of documents $d_1, \ldots, d_l$ are known and the relevance of the remaining documents $d_{l+1}, \ldots, d_n$ are unknown. The goal of a manifold-based algorithm is to predict the relevance values of the unknown documents. A manifold-based algorithm can work with the following steps to make predictions for the unknown docu-

ments:

i. From the affinity matrix $\mathbf{W}$ with its (i, j)-element defined as $w_{ij} = \exp\{-\frac{||d_i - d_j||^2}{2\sigma^2}\}$ if $i \neq j$ and $w_{ii} = 0$. Here $||d_i - d_j||$ is the distance between documents d_i and d_j.

ii. Construct the matrix $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$ in which $\mathbf{D}$ is a diagonal matrix with its (i, i)-element equal to the sum of the i-th row of $\mathbf{W}$.

iii. Iterate $\mathbf{f}(t + 1) = \alpha\mathbf{S}\mathbf{f}(t) + (1 - \alpha)\mathbf{y}$ until convergence, where α is a parameter in $(0, 1)$. Here, $\mathbf{f}$ corresponds to the relevance values of documents $\{d_1, \ldots, d_l, d_{l+1}, \ldots, d_n\}$, and $\mathbf{y}$ corresponds to the original relevance values of the documents. $\mathbf{f}(0) = \mathbf{y}$.

After the iteration converges in step iii, the documents $\{d_1, \ldots, d_l, d_{l+1}, \ldots, d_n\}$ can be reranked according to their relevance values indicated in the final $\mathbf{f}$. More details about manifold-based algorithms can be found in (Zhou et al., 2004).

2.3.5 Search result diversification

Search result diversification is similar to ad hoc search, but differs in its judging criteria and evaluation measures (Clarke and Craswell, 2011; Clarke et al., 2012). The basic premise in search result diversification is that the relevance of a set of documents depends not only on the individual relevance of its members, but also on how they relate to one another (Agrawal et al., 2009). Ideally, users can find at least one relevant document to the underlying information need. Most previous work on search result diversification can be classified as either *implicit* or *explicit* (Santos et al., 2010a,b).

Implicit approaches to result diversification promote diversity by selecting a document that differs from the documents appearing before it in terms of vocabulary, as captured by a notion of document similarity, such as cosine similarity or Kullback-Leibler divergence. Carbonell and Goldstein (1998) propose the *maximal marginal relevance* (MMR) method, which reduces redundancy while maintaining query relevance when selecting a document. Chen and Karger (2006) describe a retrieval method incorporating negative feedback in which documents are assumed to be non-relevant once they are included in the result list, with the goal of maximizing diversity. Zhai et al. (2003) present a subtopic retrieval model where the utility of a document in a ranking is dependent on other documents in the ranking and documents that cover many different subtopics of a query topic are found. Other implicit work includes, e.g., (Abbar et al., 2013) where set-based recommendation of diverse articles is proposed. In Chapter 7 below, we also tackle the problem of search result diversification implicitly, but in a different way, i.e., by data fusion.

Explicit approaches to diversification assume that a set of query aspects is available and return documents for each of them. Past work has shown that explicit approaches are usually somewhat superior to implicit diversification techniques (Vargas et al., 2012). Well-known examples include xQuAD (Santos et al., 2010a), RxQuAD (Vargas et al., 2012), IA-select (Agrawal et al., 2009), PM-2 (Dang and Croft, 2012), and, more recently, DSPApprox (Dang and Croft, 2013), text-based measures (Bache et al., 2013), term-level (Dang and Croft, 2013), learning for diversification (Zhu et al., 2014), and

fusion-based (Liang et al., 2014a). Instead of modeling a set of aspects implicitly, these algorithms obtain the set of aspects either manually, e.g., from aspect descriptions (Clarke and Craswell, 2011; Clarke et al., 2012), or they create them directly from, e.g., suggested queries generated by commercial search engines (Dang and Croft, 2012; Santos et al., 2010a) or predefined aspect categories (Szpektor et al., 2013). In Chapter 7, we propose an implicit fusion-based diversification model where we do not assume that the aspects of the query are available but do assume that we can infer the underlying topics and the prior relevance of each topic for search result diversification.

Two main components, viz., personalized web search and search result diversification, play important roles in tackling the problem of personalized search result diversification. The task of personalized web search aims at identifying the most relevant search results for an individual by leveraging their information. Many personalized web search methods have been proposed, such as the one based on social tagging profiles (Vallet et al., 2010), ranking model adaption for personalized search (Wang et al., 2013), search personalization by modeling the impact of users' behavior (Bennett et al., 2012), and personalized search using interaction behaviors in search sessions (Liu et al., 2012a). In contrast, diversification aims to make the search results diversified given an ambiguous query so that users can find at least one of these results to be relevant to their underlying information need (Agrawal et al., 2009).

Radlinski and Dumais (2006) and Vallet and Castells (2012) have studied the problem of combining both personalization and diversification. Radlinski and Dumais (2006) analyze a large sample of individual users' query logs from a web search engine such that individual users' query reformulations can be obtained. Then they personalize web search by reranking some top results using query reformulations to introduce diversity into those results. Their evaluation suggests that using diversification is a promising method to improve personalized reranking of search results. Vallet and Castells (2012) present a number of approaches that combine personalization and diversification components. They investigate the introduction of the user as an explicit variable in state-of-the-art diversification models. Their personalized search result diversification algorithms achieve competitive performance and improve over plain personalization and plain diversification baselines.

All of the previous personalized diversification models are unsupervised. However, we argue that to enhance the performance, it is better to employ a supervised learning approach, and our experiments in Chapter 8 show that supervised learning can indeed improve the performance of unsupervised approaches.

Our proposed diversified data fusion method introduced in Chapter 7 is built on a well-known search result diversification, i.e., PM-2 (proportionality model for search result diversification) (Dang and Croft, 2012). Thus, we detail the PM-2 algorithm in this section.

The main search result diversification baseline used in Chapter 7 is the proportionality-based model, which we abbreviate as PM-2 (Dang and Croft, 2012). Inspired by the scheme of the Sainte-Laguë method for assigning seats to members of competing political parties, PM-2 regards positions in the ranked list as seats, aspects of ambiguous query as competing political parties, and the weights of the aspects as the weights of votes received by the parties. The target of PM-2 is to fill in k empty positions one by one with candidate documents and then returns as ranked list L_f. To decide which doc-

ument should occupy the top empty position in the list L_f, PM-2 first recomputes the current quotient $qt[z|q]$ for each topic z given q by:

$$qt[z|q] = \frac{v_{z|q}}{2s_{z|q} + 1}, \tag{2.10}$$

where $v_{z|q}$ is the probability of topic z given q, i.e., the weight of topic z. According to the Sainte-Laguë method, the position under consideration at this step should be awarded to the topic with the largest quotient in order to best maintain the proportionality of the list. Therefore, PM-2 assigns the current position to the topic z^* with the largest quotient. The document to fill this position is the one that is not only relevant to z^* but to other topics as well:

$$\begin{aligned} d^* \quad = \quad & \arg\max_{d \in R} \big(\lambda \times qt[z^*|q] \times P(d|z^*, q) + \\ & (1-\lambda)\textstyle\sum_{z \neq z^*} qt[z|q] \times P(d|z, q)\big), \end{aligned} \tag{2.11}$$

where $P(d|z, q)$ is the probability of d talking about topic z for a given q. After document d^* is selected as diversification result, it cannot be considered again in the next process and PM-2 increases the "portion" of positions occupied by each of the topics z by its normalized relevance to d^*:

$$s_{z|q} \leftarrow s_{z|q} + \frac{P(d^*|z, q)}{\sum_{z'} P(d^*|z', q)}.$$

The above process repeats until k documents have been appended and occupied in L_f or there are no candidate documents available for the next process. The order in which a document is appended to L_f determines its ranking, and then PM-2 returns documents with this order as the diversification results for the input ambiguous query. More details about PM-2 can be found in (Dang and Croft, 2012, 2013).

In addition, a natural and direct way of diversifying a result list in the setting of data fusion is this: first rank the documents in the component lists by their estimated relevance to the query through a standard data fusion method, such as CombSUM, and then diversify the ranking through effective search result diversification models, such as MMR (Carbonell and Goldstein, 1998) and PM-2 (Dang and Croft, 2012). In our experiments in Chapter 7, we implement two more baselines, called CombSUMMMR and CombSUMPM-2. They first use CombSUM to obtain a fused list and then use MMR and PM-2, respectively, to diversify the list. We also directly utilize other plain data fusion, e.g., CombSUM and CombMNZ, to make search result diversification as data fusion methods can improve retrieval performance in terms of traditional relevance-oriented metrics like MAP and precision@k over the methods used to generate the individual result lists being fused.

2.3.6 Topic modeling

Topic models have been proposed for reducing the high dimensionality of words appearing in documents into low-dimensional "latent topics." From the first work on topic models (Hofmann, 1999), the Probablistic LSI model, topic models have received significant attention (Blei et al., 2003; Griffiths and Steyvers, 2004; Jin et al., 2011) and

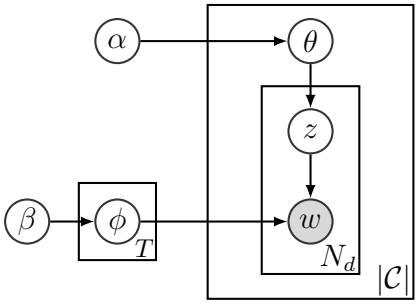


Figure 2.2: The graphical model representation of the LDA model.

have proved to be effective in many information retrieval tasks (Kurashima et al., 2013; Wei and Croft, 2006; Xu et al., 2012). Latent Dirichlet allocation (LDA) (Blei et al., 2003) represents each document as a finite mixture over "latent" topics where each topic is represented as a finite mixture over words occurring in that document. Based on LDA, many extensions have been proposed, e.g., to handle users' connections with particular documents and topics (Rosen-Zvi et al., 2004), to learn relations among different topics (Lafferty and Blei, 2005; Li and McCallum, 2006), for topic over time (Wang and McCallum, 2006), for dynamic mixture model (Wei et al., 2007), or tweet summarization (Ren et al., 2013). LDA has also been extended to sentiment analysis (Li et al., 2010).

Our proposed topic models used in Chapters 7 and 8 are based on LDA. To help understand our proposed topic models easily, here we provide the basic idea of LDA. A graphical model representation of the LDA model is shown in Fig. 2.2. In LDA, documents in the collection are represented as random mixtures over latent topics, and each topic is characterized by a multi-normal distribution over tokens appearing in the collection. The LDA can be viewed as a generative process, which can be described as follows.

i. Draw T multinomials ϕ_z from a Dirichlet prior β, one for each topic z;

ii. For each document $d \in \mathcal{C}$ in the collection with N_d tokens, draw a multinomial θ_d from a Dirichlet prior α; then for each toke w_{di} in document d:

   (a) Draw a topic z_{di} from multinomial θ_d;

   (b) Draw a token w_{di} from multinomial $\phi_{z_{di}}$;

As shown in the above process, the posterior distribution of topics depends on the information from the text. The parameterization of the LDA model is as follows:

$$\begin{aligned} \theta_d|\alpha &\sim \text{Dirichlet}(\alpha) \\ \phi_z|\beta &\sim \text{Dirichlet}(\beta) \\ z_{di}|\theta_d &\sim \text{Multinomial}(\theta_d) \\ w_{di}|\phi_{z_{di}} &\sim \text{Multinomial}(\phi_{z_{di}}) \end{aligned}$$

The parameters in the LDA model can be estimated using variational inference with the expectation-maximization algorithm (Blei et al., 2003); or an alternative inference technique uses Gibbs sampling (Griffiths and Steyvers, 2004).

2.3.7 Structured learning

Structured learning has provided principled techniques for learning structured-output models, with structured support vector machines (SSVMs) being one of the most important ones (Tsochantaridis et al., 2005). In structured learning, a set of N training pairs, $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \in \mathcal{X} \times \mathcal{Y} : i = 1, \ldots, N\}$, is assumed to be available to the learning algorithm, and the goal is to learn a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$, such that a regularized task-dependent loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ can be minimized, where $\Delta(\mathbf{y}^{(i)}, \mathbf{y})$ denotes the cost of predicting output $\mathbf{y}$ when the correct prediction is $\mathbf{y}^{(i)}$. We will use SSVMs to tackle the challenge of personalized diversification in Chapter 8. The optimization problem in structured learning can be formulated as:

Optimization Problem. (Standard structured SVMs)

$$\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i \tag{2.12}$$

subject to $\forall i, \forall \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}^{(i)}, \xi_i \geq 0$,

$$\mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \geq \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{y}) + \Delta(\mathbf{y}^{(i)}, \mathbf{y}) - \xi_i.$$

In the objective function (2.12), the parameter C is a tradeoff between model complexity, $||\mathbf{w}||^2$, and a hinge loss relaxation of the training loss for each training example, $\sum \xi_i$. The constraints enforce the requirement that the ground-truth $\mathbf{y}^{(i)}$ should have a greater function value than other alternative $\mathbf{y} \in \mathcal{Y}$, and $\mathbf{y} \neq \mathbf{y}^{(i)}$. The optimization problem defined in (2.12) can be solved by employing the cutting plane algorithm (Tsochantaridis et al., 2005).

In the past few years, Structured SVMs (SSVMs) have been studied and applied in many areas, such as speech recognition (Zhang and Gales, 2013), optimizing average precision of a ranking (Yue et al., 2007), and diversification (Yue and Joachims, 2008). For us, the most interesting prior application of SSVMs is the one for predicting diverse subsets (Yue and Joachims, 2008). However, our personalized search result diversification method differs from that proposed in (Yue and Joachims, 2008): we work on personalized diversification where we propose a user-interest LDA-style model in Chapter 8 to capture a user's interest distribution over topics, whereas they directly apply existing SSVMs algorithm to tackle the problem of search result diversification but not personalized diversification; our model explicitly makes results diverse and consistent to the user's interest by enforcing both diversity and interest constraints, whereas their model only implicitly diversifies the results by adopting standard SSVMs. Prior work on diversification (Dang and Croft, 2012; Santos et al., 2010a; Vargas et al., 2012), however, has shown that explicit approaches outperform implicit ones in most cases.

This chapter has described the tasks and corresponding methods. In the next chapter we will describe our experimental methodology and setup.

3 Experimental Methodology

In previous chapter we have introduced the background for our two main research topics, data fusion and search result diversification. We now move on to the experimental methodology that is shared by the research chapters (Chapters 4–8). To begin with, we first give an introduction of test collections and retrieval evaluation in IR in Section 3.1. Then, we describe the test collections used in the research chapters (Chapters 4–8) in Section 3.2. We detail the retrieval evaluation metrics used to evaluate the search results in Section 3.3. Finally, we detail significance testing techniques used to make comparisons between our proposed methods and the baselines in Section 3.4.

3.1 Introduction

The availability of test collections plays a critical role in successful experimental research in IR. There are a number of academic communities providing test collections to evaluate the performance of retrieval algorithms. Among these, the best known ones include the Text REtrieval Conference (TREC),[1] the Cross-Language Evaluation Forum (CLEF),[2] the NII Test Collection for IR systems project (NTCIR),[3] and the Forum for Information Retrieval Evaluation (FIRE).[4] In this thesis, we mainly use datasets provided by TREC as well as runs submitted to TREC to perform our experiments to evaluate our proposed data fusion models and the baseline models in Chapters 4–6. There are a couple of reasons that we use the publicly available data in the TREC: the input lists to evaluate our proposed model can also be reused by other models, which indeed allows other researchers to make comparisons as the evaluation can be performed with the exact same datasets as well as the same input result lists. Also, in Chapter 8 for the purpose of making comparisons between our proposed personalized diversification algorithm and other existing algorithms, we prefer to use a publicly downloadable dataset.

Additionally, the availability of *retrieval evaluation metrics* also plays a critical role in the evaluation of retrieval systems in IR. Retrieval evaluation is a process of systematically associating quantitative metrics to the results produced by an IR system in response

[1] http://trec.nist.gov
[2] http://hmi.ewi.utwente.nl/Projects/clef.html
[3] http://research.nii.ac.jp/ntcir
[4] http://www.isical.ac.in/~clia/

to a set of user queries (Baeza-Yates and Ribeiro-Neto, 2011). Retrieval evaluation metrics should be directly associated with the relevance of the results to the users. A common approach to compute scores of metrics is to compare the results produced by the search system with results suggested by humans for the same set of queries. Here, the retrieval evaluation metrics concern the quality of the results, not the performance of the system, such as how fast it processes queries.

In the next section (Section 3.2) we provide details of the test collections used in the research chapters (Chapters 4–8).

3.2 Test Collections Used in the Thesis

3.2.1 Ad hoc search collection

In order to answer our research questions about how to utilize manifold-based algorithm for ad hoc search in Chapter 6, we work with three text collections provided by the ad hoc track of TREC-3 (Harman, 1994), the web track of TREC-10 (Hawking and Craswell, 2002) and the robust retrieval track of TREC-12 (Voorhees, 2005), respectively. The collection provided by the ad hoc track of TREC-3 is comprised of 50 queries (topics 151–200), and 741,856 documents (news and journal articles); the data for the web track of TREC-10 is the 10 gigabyte WT10g (Hawking and Craswell, 2002) collection, with 50 queries (topics 501–550) and 1,692,096 documents; and the data for the robust retrieval track of TREC-12 consists of 50 queries and 528,155 documents on TREC disks 4 and 5 minus the *Congressional Record*. The tasks studied at these three tracks were: new questions (queries) were assumed to be asked against a set of data, and the system had to retrieve relevant documents to the questions. Participants produced 40, 97 and 78 submitted runs at TREC-3, TREC-10 and TREC-12 for the tracks, respectively. The p@20 performance of the runs varies from 0.0620 to 0.6740, from 0.0010 to 0.4730, and from 0.0895 to 0.3930 in the TREC-3, TREC-10 and TREC-12, respectively. Some details about the implementations of the runs in the tracks can be found in (Harman, 1994; Hawking and Craswell, 2002; Voorhees, 2005).

3.2.2 Microblog search collection

In order to answer our research questions about how to utilize data fusion for searching posts in a microblogging environment in Chapters 4–5, we work with the Tweets 2011 corpus (Macdonald et al., 2011), called Tweet11, provided by the TREC 2011 Microblog track. The collection is comprised of approximately 16 million tweets collected over a period of 2 weeks (23th January until 8th February 2011, inclusive) sampled courtesy of Twitter. Different types of tweets in this data set are present, including replies and retweets. Each tweet has its own timestamp. Descriptive statistics about the collection are provided in Table 3.1.

The task studied at the TREC 2011 Microblog track was: given a query with a timestamp, return relevant and interesting tweets in reverse chronological order. This task is akin to ad hoc search on Twitter, where a user's information need is represented by a query at a specific time. For 2012, the setting of the track was almost the same as that in

Table 3.1: Description of the data set used in our experiments.

Number of tweets	15,137,399
Number of users	4,670,516
Median tweet length	8.66
Median English tweet length	10.76
Number of English tweets	9,318,772
Number of English retweets	1,069,006
Number of hyperlinks	1,135,720
Number of hashtags	1,005,343

2011 except that the topics were different and the result lists were required to be ordered by relevance instead of chronologically (Soboroff et al., 2012). In our experiments, we rank tweets by relevance.[5]

We use two sets of test topics (queries) in our experiments, the 2011 test set and the 2012 test set. In total, NIST (the National Institute of Standards and Technology) created 50 test topics for the TREC 2011 Microblog track, each representing an information need at a specific point in time when the topics were issued. 49 test topics were used in the TREC and 2965 tweets were deemed relevant; some topics have just two relevant tweets while some have more than 100 relevant tweets. Indeed, one of the 50 topics originally created, MB050, did not have any relevant tweets in the pool, and it was therefore dropped from the evaluation. To assess the tweets, the assessors judged the relevance of a tweet after reading it. Tweets in the Tweet11 corpus were judged on the basis of the defined information need using a three-point scale: *Not Relevant*, *Minimally Relevant* and *Highly Relevant*.

A total of 59 groups participated in the TREC 2011 Microblog track, with each team submitting at most four runs, which resulted in 184 runs[6] (Lin et al., 2012; Macdonald et al., 2011). The official evaluation metric was precision at 30 (p@30) (Macdonald et al., 2011). The p@30 scores of these 184 runs varied dramatically, with the best run achieving a p@30 score of 0.4551 and the worst run achieving 0.000. In our experiments below, we do not use any runs whose p@30 scores are below 0.10, leaving us with 174 runs from the TREC 2011 Microblog track. Details about the implementation of each run from the TREC 2011 Microblog track can be found in (Lin et al., 2012; Macdonald et al., 2011).

The Microblog search track continued in 2012 using the same corpus, Tweet11 (Soboroff et al., 2012). NIST created 60 new test topics representing information needs at specific points in time in TREC 2012 and labeled 6286 tweets as minimally or highly relevant. The TREC 2012 Microblog track received 121 runs[6] from 33 participating groups. The best run in TREC 2012 Microblog track is hitURLrun3 (Han et al., 2012), with its p@30 score being 0.4695. Once again, in our experiments, we only use the runs whose p@30 scores are no less than 0.10, leaving us with 117 runs from the TREC 2012 Microblog track. For details about the implementation of the runs from the TREC 2012 Microblog track we refer to (Soboroff et al., 2012). The track continued in 2013, but

[5] We reorder all 2011 runs by retrieval score before fusing them.

[6] The 2011 and 2012 runs can be downloaded from `http://trec.nist.gov`.

with an incomparable setup.

3.2.3 Web track collections

In order to answer our research questions about how to enhance the retrieval performance of diversification in Chapter 7 we work with the runs submitted to the TREC 2009, 2010, 2011 and 2012 Web tracks, and the billion-page ClueWeb09 collection.[7] The ClueWeb09 dataset was created to support research in IR. It consists of about one billion web pages in ten languages that were collected in January and February 2009. There are two tasks in these tracks: an ad hoc search task and a search result diversification task (Clarke and Craswell, 2011; Clarke et al., 2009, 2010, 2012). We only focus on the diversification task, where the top-k documents returned should not only be relevant but also cover as many aspects as possible in response to a given query. In total, we have 200 ambiguous queries from the four years, with 2 queries (#95 and #100 in the 2010 edition) not having any relevant documents. Typically, each query has 2 to 5 aspects, and some relevant documents are relevant to more than 2 aspects of the query.

Many of the runs submitted to these four years of the Web track for the diversification task were generated by state-of-the-art diversification methods. In total, we have 119, 88, 62 and 48 runs from the 2009, 2010, 2011 and 2012 editions, respectively.[8]

3.2.4 Personalized diversification collection

In order to answer our research questions about how to enhance retrieval performance of personalized diversification in Chapter 8, we work with a publicly available personalized diversification dataset.[9] This dataset contains private evaluation information from 35 users on 180 search queries. The queries are quite ambiguous, as the length of each query is no more that two keywords. In total, there are 751 subtopics for the queries, with most of the queries having more than 2 subtopics. Over 3,800 relevance judgements are available, for at least the top 5 results for each query. Each relevance judgement includes 3 main assessments: a 4-grade scale assessment on how relevant the result is to the user's interests (resulting in the *user relevance* ground truth and the set of users' interesting documents being created); a 4-grade scale assessment on how relevant the result is to the evaluated query (resulting in the *topic relevance* ground truth being created); and a 2-grade assessment whether a specific subtopic is related to the evaluated query (resulting in the subjective subtopics related to the search query being created). Details of this dataset can be found in (Vallet and Castells, 2012). For pre-processing, we apply Porter stemming, tokenization, and stopword removal (using the INQUERY list) to the documents using the Lemur toolkit.[10]

Two well-known corpora, ClueWeb09 and ClueWeb12,[11] have been proposed for the search result diversification tasks in the TREC 2009–2013 Web tracks (Clarke et al.,

[7] Available from `http://boston.lti.cs.cmu.edu/Data/clueweb09`.
[8] All runs are available from `http://trec.nist.gov`.
[9] `http://ir.ii.uam.es/~david/persdivers/`
[10] `http://www.lemurproject.org`
[11] `http://boston.lti.cs.cmu.edu/clueweb12/`

2012). However, they do not contain any user information or relevance judgments provided by specific users, and thus do not fit our experiments in Chapter 8.

3.3 Evaluation Metrics

Evaluation is an import aspect of our methodology. In this section we first detail the evaluation metrics that we use to assess the performance of our proposed retrieval models and the baselines. Then we briefly introduce the significance testing we perform to compare retrieval results between different models.

To evaluate the effectiveness of both our data fusion and search result diversification models in Chapters 4–8, we use a set of common IR evaluation metrics (Baeza-Yates and Ribeiro-Neto, 2011; Manning et al., 2008), like p@k, and also those proposed in recent years, like Prec-IA@k. We distinguish the evaluation metrics used in this thesis into three types: (i) for the task of ad hoc search; (ii) for the task of microblog search task; and (iii) for the task of search result diversification.

3.3.1 Metrics for ad hoc search

p@k. The precision at rank k (p@k) metric indicates the percentage of relevant documents within the top k returned documents. In web search related tasks this metric is often considered important, as users usually only look at top k returned documents of a ranked list. It can be easily calculated as:

$$p@k = \frac{\sum_{r=1}^{k} \mathrm{rel}(r)}{k}, \tag{3.1}$$

where $\mathrm{rel}(r)$ is a binary function that indicates whether or not the document at rank r is relevant:

$$\mathrm{rel}(r) = \begin{cases} 1 & \text{if } r \in \mathcal{R} \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where $\mathcal{R}$ is the set of relevant documents for a given query.

MAP. Mean average precision (MAP) is a metric that measures both precision and recall of search results and is used most commonly in research in the field of IR. For each relevant document in the returned document list we take the precision at the position of that document. We sum over these precision values and divide it by the total number of relevant documents. This gives us the average precision (AP) for a query:

$$AP = \frac{\sum_{k=1}^{N} p@k \cdot \mathrm{rel}(k)}{|\mathcal{R}|}, \tag{3.3}$$

where N is the number of returned documents (in most TREC tasks N=1,000). When we take the mean of AP values over a set of test queries, we get the mean average precision (MAP) for a system on that set of queries. In other words, the MAP score of a retrieval system can be obtained by averaging the AP values of all the test queries.

nDCG. Given a ranked result set of documents (in our setting, groups) $\mathcal{S}$ and an ideal ordering of the same set of documents $\mathcal{O}$, the *discounted cumulative gain* (DCG) (Järvelin and Kekäläinen, 2002) at a particular rank threshold k is defined as:

$$\mathrm{DCG}(\mathcal{S}, k) = \sum_{j=1}^{k} \frac{2^{r(j)} - 1}{\log(1 + j)}, \tag{3.4}$$

where $r(j)$ is the judgment ($0 =$ Bad, $1 =$ Fair, $2 =$ Good, $3 =$ Excellent, etc.) at rank j in set $\mathcal{S}$. The ideally ordered set $\mathcal{O}$ contains all documents rated for the given query sorted descending by the judgment value. Then the *normalized discounted cumulative gain* (nDCG) (Järvelin and Kekäläinen, 2002) at a particular rank threshold k is defined as:

$$\mathrm{nDCG}(\mathcal{S}, k) = \frac{\mathrm{DCG}(\mathcal{S}, k)}{\mathrm{DCG}(\mathcal{O}, k)}. \tag{3.5}$$

nDCG discounts the contribution of a document to the score as its rank increases. Higher nDCG values correspond to better correlation with human judgments. nDCG value at rank threshold k when the set $\mathcal{S}$ is clear from the context is often written as $\mathrm{nDCG@}k$.

Diversity-oriented metrics used in this thesis includes α-nDCG@k, S-Recall@k, Prec-IA@k and MAP-IA@k, all of which are official evaluation metrics in the search result diversification of the TREC Web tracks (Clarke and Craswell, 2011; Clarke et al., 2009, 2010, 2012).

3.3.2 Metrics for microblog search

Relevance-oriented metrics used in this thesis include p@k, MAP, nDCG. One of the reasons we consider these metrics is to make our data fusion models comparable to previous models. For performance evaluation in our experiments of data fusion models in Chapters 4–5 we consider both minimally relevant and highly relevant posts and use the official metric, p@30. We also report on p@5, p@10, p@15 and MAP scores. We use trec_eval[12] to compute the performance scores. We expect our proposed data fusion models to have a recall-enhancing effect. This may negatively impact very early precision, which is why we include p@5. But we hypothesize that we will see an increase in precision scores at lower ranks because of the expected boost in recall and the limited length of the lists being scored (only 30 items). For this reason we consider precision scores at multiple cut-offs (5, 10, 15, 30) as well as MAP. As some documents are denoted as relevant or highly relevant to the queries, we also consider nDCG (Clarke et al., 2008a) to measure the performance if necessary.

3.3.3 Metrics for search result diversification

α-nDCG@k. A version of normalized discounted cumulative gain at k in which the role of the parameter α is emphasized in computing the novelty of the top k documents.

[12]The evaluation script can be obtained from `http://trec.nist.gov`.

α-nDCG@k scores a ranking by rewarding newly-found subtopics and penalizing redundant subtopics geometrically, discounting all rewards with a log-harmonic discount function of rank. See (Clarke et al., 2008b) for details on how α-nDCG@k is computed.

S-Recall@k. Subtopic recall at k (Zhai et al., 2003) is computed at retrieval depth k using the following procedure. Assume there are Q ambiguous queries. Let z be an aspect of query q and N_q the number of aspects (subtopics) associated with q. Then, the subtopic recall at rank k (Zhai et al., 2003) is defined as the percentage of subtopics covered by one of the top k documents:

$$\text{S-Recall@}k = \frac{1}{Q}\sum_{q=1}^{Q}\frac{|\bigcup_{i=1}^{k}\text{subtopics}(d_i|q)|}{N_q}, \tag{3.6}$$

where subtopics$(d_i|q)$ is the number of aspects covered by d_i in response to q.

ERR-IA@k. Intent-aware expected reciprocal rank at retrieval depth k, similarly, is computed as:

$$\text{ERR-IA@}k = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{N_q}\sum_{z=1}^{N_q}\text{ERR}(k|z,q), \tag{3.7}$$

where ERR$(k|z,q)$ is the expected reciprocal rank score at k in terms of aspect z of query q.

Prec-IA@k. Intent-aware precision at k (Agrawal et al., 2009) is defined as:

$$\text{Prec-IA@}k = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{N_q}\sum_{z=1}^{N_q}\text{Prec}(k|z,q), \tag{3.8}$$

where Prec$(k|z,q)$ is the precision at k in terms of the aspects z of q, and can be computed as $\frac{1}{k}\sum_{j=1}^{k} j_q(z,j)$. Here, $j_q(z,j) = 1$ if the document returned for q at depth j is judged relevant to aspect z of q; otherwise, $j_q(z,j) = 0$.

MAP-IA@k. Intent-aware MAP at k (Agrawal et al., 2009) is computed as:

$$\text{MAP-IA@}k = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{N_q}\sum_{z=1}^{N_q}\text{MAP}(k|z,q), \tag{3.9}$$

where MAP$(k|z,q)$ is the MAP score for top k returned documents in terms of aspect z of q.

3.4 Significance Testing

Researchers in IR commonly use three types of statistical significance test, i.e., Student's paired t-test (Gosset, 1904), the Wilcoxon signed rank test (Wilcoxon, 1945) and the sign test (Karas and Savage, 1967). The reasons for using significance testing is that we want to determine whether observed differences may be due to chance. Other reasons include, for instance, there is inherent noise in an evaluation. Some topics are harder than others, and the assessors hired to judge relevance of documents are human and thus open to variability in their behavior (Smucker et al., 2007).

In Chapters 4–8 we introduce approaches that should improve performance on the tasks in these chapters. To test if our proposed approaches really do show improvements

we compare their scores to baseline scores. These baseline scores indicate how the systems developed by our proposed approaches perform. When comparing two runs, we want to test for significant differences between them. To this end we commonly use a two-tailed paired t-test. Gosset (1904) shows that in practice there is no difference between the t-test and the randomization test, although the latter is a more principled choice. In this thesis we opt for the t-test as we want to promote retrieval models that truly are better than other models that by chance performed better given the set of topics, judgments, and documents used in the evaluation.

Statistically significance of observed differences between the performance of two runs for $\alpha = 0.01$ and $\alpha = 0.05$ is shown in the tables of this thesis when necessary and appropriated, the former being stronger than the latter. Results marked by $^{\blacktriangle}$and $^{\blacktriangledown}$reflect statistically significant improvements or drops for $\alpha = 0.01$ and $^{\triangle}$and $^{\triangledown}$do the same for $\alpha = 0.05$.

Until now, we have introduced the background and experimental methodology of this thesis. From next chapter, we will start our investigation on data fusion and search result diversification.

4

Burst-Aware Data Fusion

In the previous two chapters, we have introduced the background material and the experimental methodology for this thesis. In this chapter, we begin our research and try to answer the research questions listed in Chapter 1. In this chapter, we explore how to improve the performance of microblog search in microblogging environments such as Twitter[1] via data fusion.

Microblogging platforms have become indispensable communication channels through which hundreds of millions of users around the world witness breaking news events. The characteristics of the posts, such as their limited length, along with easy access on many platforms, lead to regular status updates by large numbers of people (Zhao and Rosson, 2009). Microblogging platforms display fast paced dynamics as reflected by rapidly evolving topics (Yang and Leskovec, 2011). Searching posts in such rapidly changing environments is a challenge (Lin et al., 2012). To tackle this problem, much previous work has focused on content-based criteria for ranking posts in response to a query, in combination with a broad range of other ranking criteria, including, e.g., the existence of hyperlinks, hashtags and retweets.

As we explained in Chapter 2, fusion is a popular method for generating result lists based on multiple ranking criteria. In this chapter, we look at the problem of searching microblog posts as a late data fusion task (Shaw and Fox, 1993): we fuse ranked lists of posts produced by a diverse set of microblog post rankers into a single final ranked list of posts. In the following, we consider the case where only ranks and publication times are available and no other additional information is provided such as the retrieval status values or the contents of the posts. We focus on a particular microblog search scenario, one that was studied at the Text REtrieval Conference (TREC) 2011 and 2012 Microblog tracks (Lin et al., 2012; Soboroff et al., 2012). The task uses Twitter data and is defined as follows: given a query with a timestamp, return relevant and interesting tweets.

In this chapter, we focus on how to improve the performance of searching posts in microblogging environment. Specifically, we seek to answer the following main research question:

RQ 1 Can data fusion help microblog search?

To answer this main research question, we first take a look at the characteristics of microblog environments. In such environments people tend to talk about a topic mostly during

[1] `http://www.twitter.com`

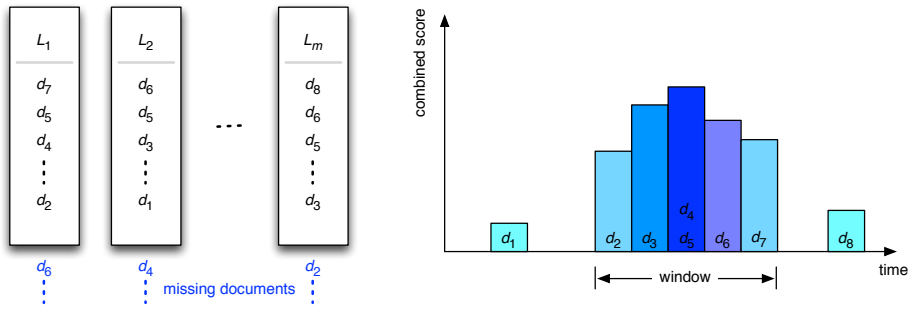


Figure 4.1: Rewarding posts that are published in the same narrow time frame as a large number of (supposedly) relevant posts. On the left, we display m ranked lists of posts that were produced in response to a given query; these lists need to be fused. Post d_2 only occurs in list L_1 and it is ranked low in L_2; d_8 also occurs in a single list, L_m, but it is ranked very high. On the right, we show the distribution of the publication timestamps of the documents in the lists to be combined. The vertical axis indicates the combined scores of posts with the same timestamps based on a baseline fusion method, e.g., CombSUM. According to its publication timestamp, d_2 was published in a "good" period for the query: many posts published around the same time as d_2 are highly ranked in many lists; because of this, BurstFuseX will "reward" d_2. In contrast, d_8 does not have a publication time around which many highly ranked posts were published, hence it should not receive a reward. Documents marked in blue are "missing" documents; e.g., d_6 is a missing document for L_1, as it is not observed in L_1 during the fusion process. See Chapter 5 for more details on missing documents.

specific short time intervals (Chen et al., 2010; Hoonlor et al., 2012; Lappas et al., 2009; Mathioudakis et al., 2010; Peetz et al., 2012; Vlachos et al., 2004). For instance, people talked about the "2014 Eastern Synchronized Skating Sectional Championship" mainly between January 30 and February 1, 2014, which is when the championship was held. Posts created before the beginning or after the ending of the event are less likely to talk about the championship competitions and, hence, are less likely to be relevant. This observation leads to the following intuition about fusing ranked lists of microblog posts.

> If a post d and (other) relevant posts $d_1, \ldots, d_k$ were published within the same narrow time window, and the relevant posts $d_1, \ldots, d_k$ are ranked highly in many of the lists to be merged, then post d should be "rewarded" by boosting its rank, even if, in the extreme case, it appears in only one list where it is ranked low.

Fig. 4.1 illustrates this intuition; there, post d_2 is ranked low in list L_1 but our intuition suggests that it should be rewarded as it was published in the same narrow time window in which a large number of posts occur that are ranked high in many lists; in contrast, d_8, while ranked high in L_m, receives no such bonus as it was published outside the narrow window.

To answer the main research question and tackle the problem of microblog post

search, we propose BurstFuseX, a novel probabilistic model that not only utilizes information traditionally used when merging ranked lists, such as ranks, but also exploits temporal information, i.e., the publication timestamps of microblog posts. In our fusion model, we focus on the case where only ranks and publication timestamps are available and no additional information is provided—such as the content of the posts, the post's RSVs (Relevance Status Values), the resources the posts link to, etc. In fact, accessing the contents of posts may be inefficient and hence inappropriate in dynamic environments such as microblog search. In addition, the content may not be available in all scenarios (Salakhutdinov and Mnih, 2008a). Briefly, BurstFuseX first calls a standard document fusion method X to merge a set of ranked lists of microblog posts for a given query. Subsequently, as illustrated in Fig. 4.1, based on the fused scores produced by fusion method X, we detect windows of timestamps of high-scoring posts. These windows give rise to bursts of posts. We then reward posts that are published in the temporal vicinity of a burst that contains high-scoring posts.

In our experiments aimed at assessing the performance of BurstFuseX, we sample runs that have been submitted to the TREC 2011 and 2012 Microblog tracks and fuse them using BurstFuseX, respectively. For the underlying fusion method X (on top of which BurstFuseX builds), we consider three alternatives: two unsupervised fusion methods, CombSUM (Shaw and Fox, 1993), CombMNZ (Lee, 1995), and one state-of-the-art supervised fusion method: λ-Merge (Sheldon et al., 2011). For further comparisons, we consider a number of burst or time-sensitive microblog retrieval baselines. As BurstFuseX detects bursts based on the output of a standard fusion method rather than on the contents of microblogs, we also consider a baseline that detects bursts based on the contents. As we will see below, BurstFuseX significantly outperforms most fusion approaches and burst or time-sensitive retrieval methods.

Our contributions in this chapter can be summarized as follows:

i. We propose a novel and effective probabilistic data fusion model to microblog post search, BurstFuseX, which not only takes traditional information such as document rank into account, but also exploits the temporal characteristics of microblog environments.
ii. To the best of our knowledge, this is the first attempt to frame the problem of searching microblog posts as a data fusion problem and also the first attempt to integrate temporal characteristics of result sets into data fusion.

In Section 4.1 we detail BurstFuseX; we follow with a description of our experimental setup in Section 4.2 and report on our experimental results and perform topic-level and run-time analyses in Section 4.3. Finally, Section 4.4 concludes the chapter.

4.1 Fusion Approach

In this section, we first provide the task we address. Then we briefly describe standard unsupervised and supervised data fusion methods that will be integrated in our proposed data fusion methods and taken as baselines in our experiments in Section 4.1.1. We define what bursts are and detail how to detect bursts in data fusion scenarios in Sec-

tion 4.1.2. After that, we detail our proposed data fusion methods for microblog search in Section 4.1.3.

The task we address in this chapter is the following: *Given a query and a set of ranked lists of posts returned in response to the query, fuse the lists into a single ranked list of posts to be returned in response to the query*. Hence, the input of our burst-aware data fusion method BurstFuseX consists of a query and a set of ranked lists of posts; the output is a single fused list. Algorithm 1 gives a high level overview of BurstFuseX.

Algorithm 1: BurstFuseX: Burst-aware data fusion for microblog post search.

Input : A query q
A number of ranked lists of posts to be fused, $L_1, L_2, \ldots, L_m$
The combined set of posts $\mathcal{C}_\mathbf{L} := \bigcup_{i=1}^{m} L_i$
A standard fusion method X.
Output: A final fused list of posts.

1 Calculate the (standard) fusion score $F_X(d; q)$ according to X for each post $d \in \mathcal{C}_\mathbf{L}$; see Section 4.1.1;
2 Detect bursts based on the timestamps and $F_X(d; q)$ scores; see Section 4.1.2;
3 Calculate the BurstFuseX fusion score for each $d \in \mathcal{C}_\mathbf{L}$ using the bursts and the standard fusion score; see Section 4.1.3;
4 Construct the final fused list based on the BurstFuseX score of $d \in \mathcal{C}_\mathbf{L}$ obtained in step 3.

In the remainder of this section we detail the steps that make up BurstFuseX. In Table 4.1 we list the notation that we use. Other notation used in this chapter can be found in Table 2.1 in Chapter 2.

The fusion methods we consider as building blocks for BurstFuseX all assign a non-negative *fusion score* $F_X(d; q)$ to every post $d \in \mathcal{C}_\mathbf{L}$. We set $F_X(d; q) := 0$ for $d \notin \mathcal{C}_\mathbf{L}$, following Bruno and Marchand-Maillet (2009); Fox and Shaw (1994); Kozorovitsky and Kurland (2011); Lee (1995); Wu (2012). The higher $F_X(d; q)$ is, the more likely d is assumed to be an appropriate response to q.

4.1.1 Standard fusion methods

To be able to define the final fusion score $F_{BurstFuseX}(d; q)$ we integrate and make use of an existing standard fusion method (step 1 of Algorithm 1). BurstFuseX is independent of the particular choice of the standard fusion method that it integrates: any fusion method can be integrated into our model. Below, we briefly review a supervised method, λ-Merge (Sheldon et al., 2011). An overview of two standard unsupervised fusion methods, CombSUM and CombMNZ, can be found in Chapter 2.

Recently, several supervised methods for merging ranked lists have been proposed, one of which is λ-Merge (Sheldon et al., 2011). In this chapter, we view λ-Merge as a typical representative of the supervised standard fusion methods that are currently available.[2]

[2]To be able to define λ-Merge, we need to assume that we can access the content of posts.

Table 4.1: Additional notation used in this chapter (cf. Table 2.1).

Notation	Gloss
k_{L_i}	length of list L_i
α_m	weight of a list; used in the definition of λ-Merge
$g(d;q)$	scoring function used in the definition of λ-Merge
$f(\boldsymbol{x};\boldsymbol{\theta})$	linear scoring function used in the definition of λ-Merge
$\mathcal{S}_{t_i}(\mathcal{C}_\mathbf{L})$	burst-time score at time t_i
$t_{\mathcal{C}_\mathbf{L}}$	number of different timestamps of posts in $\mathcal{C}_\mathbf{L}$
$\Im(\mathcal{C}_\mathbf{L})$	sequence of burst-time scores
$b(\mathcal{C}_\mathbf{L})[t_i:t_j]$	burst with start timestamp t_i and end timestamp t_j (given query q), abbreviated by b
$\mathcal{B}(\mathcal{C}_\mathbf{L})$	set of all bursts in $\mathcal{C}_\mathbf{L}$ (given query q)
μ	free parameter that governs burst information
σ_b	standard deviation of timestamps belonging to the burst b

Given a query, λ-Merge can directly optimize a retrieval metric (e.g., MAP) to enhance retrieval effectiveness under the assumption that query reformulation candidates are available. In particular, λ-Merge learns a scoring function to rank documents from multiple reformulations of the given query by combining features that indicate document quality (such as retrieval score) with features that indicate the quality of the reformulation and its results lists (called *gating features* in (Sheldon et al., 2011)). In our setting, we do not assume that query reformulation candidates are easily available, i.e., no features about the quality of the reformulation (*gating features*) are used in our data fusion method.

Our settings for λ-Merge are detailed in an appendix to the chapter (see Appendix 4.A).

4.1.2 Bursts and burst detection

To ground our intuitions about utilizing burst information to boost the performance of microblog search, we choose four test queries as examples and examine plots of the number of relevant documents distributed over their document ages (measured by days) in Fig. 4.2.[3] The figure confirms that people tend to talk about topics within specific time windows. It is, therefore, worthwhile to detect such time windows ("bursts") and to use such burst information, which is what our proposed data fusion method for microblog search aims to do.

Next, we move on to the next step (step 2) of Algorithm 1 and detail how we detect bursts. Let t_i be a timestamp. Let d_{t_i} ($\in \mathcal{C}_\mathbf{L}$) denote a post d with timestamp t_i. We regard posts published during the same hour as having the same timestamp. Although it is possible to define "the same timestamp" in many different ways, we found that this is a suitable level of granularity for the fusion effectiveness of searching posts; the same setting is also used in (Metzler et al., 2012). Now, before we detect bursts, we need to define $\mathcal{S}_{t_i}(\mathcal{C}_\mathbf{L})$, the *burst-time score at time t_i of $\mathcal{C}_\mathbf{L}$*, the set of posts occurring in the lists under consideration. Let $f_X(d_{t_i};q)$ be the score of d_{t_i} given q under the standard fusion

[3]The topics are selected from test collections detailed in Section 3.2.2.

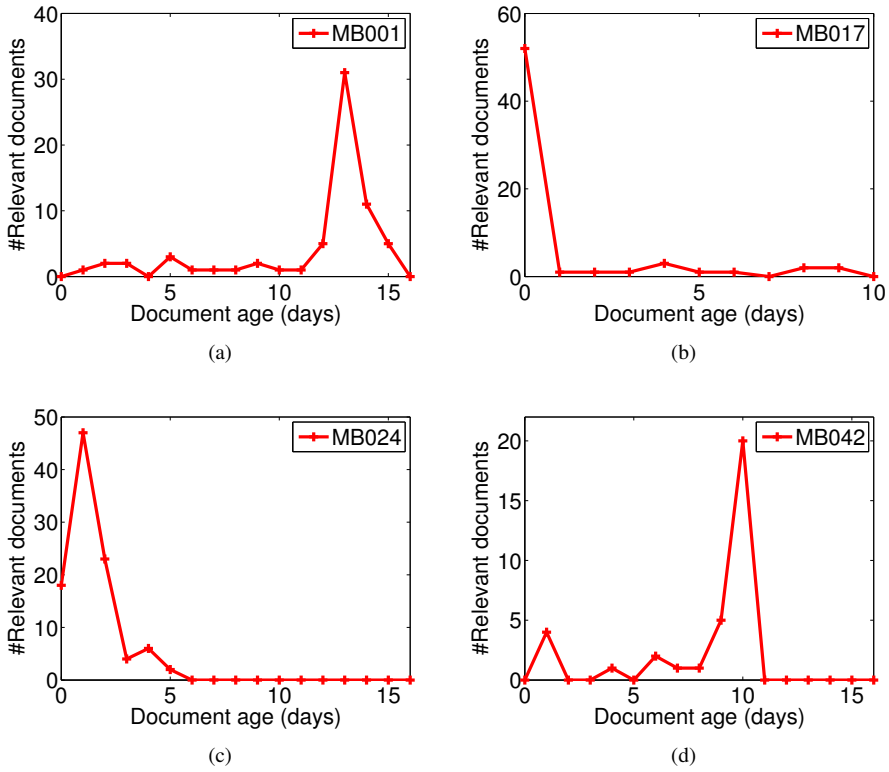


Figure 4.2: Distribution of the number of relevant documents over days for four test queries. In each subfigure, the x-axis indicates document ages from query time to the document timestamps, and the y-axis indicates the number of relevant documents according to the ground-truth in the Tweets 2011 dataset (detailed in Section 3.2.2). Subfigure (a) plots the relevant documents over time for query MB001 – *BBC World Service staff cuts*, (b) is for MB017 – *White Stripes breakup*, (c) is for MB024 – *Super Bowl, seats*, and (d) is for MB042 – *Holland Iran envoy recall*.

method X. Then:

$$\mathcal{S}_{t_i}(\mathcal{C}_\mathbf{L}) = \frac{\sum_{d_{t_i} \in \mathcal{C}_\mathbf{L}} f_X(d_{t_i}; q)}{\sum_{j=1}^{t_{\mathcal{C}_\mathbf{L}}} \sum_{d_{t_j} \in \mathcal{C}_\mathbf{L}} f_X(d_{t_j}; q)} - \frac{1}{t_{\mathcal{C}_\mathbf{L}}}, \qquad 1 \leq i \leq t_{\mathcal{C}_\mathbf{L}} \tag{4.1}$$

where $1 \leq j \leq t_{\mathcal{C}_\mathbf{L}}$ and $t_{\mathcal{C}_\mathbf{L}}$ is the total number of different timestamps belonging to posts in $\mathcal{C}_\mathbf{L}$. Notice that the burst-time score $\mathcal{S}_{t_i}(\mathcal{C}_\mathbf{L}) > 0$ if it is above the average score (i.e., $1/t_{\mathcal{C}_\mathbf{L}}$), otherwise $\mathcal{S}_{t_i}(\mathcal{C}_\mathbf{L}) \leq 0$.

We compute a burst-time score $\mathcal{S}_{t_i}(\mathcal{C}_\mathbf{L})$ at each time point $t_i \in \{t_1, t_2, \ldots, t_{\mathcal{C}_\mathbf{L}}\}$ in $\mathcal{C}_\mathbf{L}$. In this manner we generate a *burst-time score sequence* $\Im(\mathcal{C}_\mathbf{L}) = \{\mathcal{S}_{t_1}(\mathcal{C}_\mathbf{L}), \mathcal{S}_{t_2}(\mathcal{C}_\mathbf{L}), \ldots, \mathcal{S}_{t_{\mathcal{C}_\mathbf{L}}}(\mathcal{C}_\mathbf{L})\}$.

Following (Ruzzo and Tompa, 1999), a segment $\Im(\mathcal{C}_\mathbf{L})[t_i : t_j] = \{\mathcal{S}_{t_i}(\mathcal{C}_\mathbf{L}), \mathcal{S}_{t_{i+1}}(\mathcal{C}_\mathbf{L}), \ldots, \mathcal{S}_{t_j}(\mathcal{C}_\mathbf{L})\}$, where $1 \leq i \leq j \leq t_{\mathcal{C}_\mathbf{L}}$, is a *maximal segment* in $\Im(\mathcal{C}_\mathbf{L})$ if:

i All proper subsequences of $\Im(\mathcal{C}_\mathbf{L})[t_i : t_j]$ have a lower score.[4]

ii No proper super-segments of $\Im(\mathcal{C}_\mathbf{L})[t_i : t_j]$ in $\Im(\mathcal{C}_\mathbf{L})$ satisfy item **i**.

We adapt a linear-time algorithm proposed in (Ruzzo and Tompa, 1999) to find *all maximal segments* in the sequence $\Im(\mathcal{C}_\mathbf{L})$. As an example, consider the input sequence $\Im(\mathcal{C}_\mathbf{L}) = \{2, -2, 4, 3, -3, -4, -1, -3, 5, -1, 3, -2\}$. The maximal segments in this sequence are $\{2\}$, $\{4, 3\}$ and $\{5, -1, 3\}$. The segment $\{2, -2, 4, 3\}$ is not maximal, since it has a nonempty zero-scoring prefix $\{2, -2\}$ appending to the left of $\{4, 3\}$; $\{5\}$ is not a maximal segment, since $\{5, -1, 3\}$ has a total higher score of 7. Each maximal segment $\Im(\mathcal{C}_\mathbf{L})[t_i : t_j]$ gives rise to a *burst* of posts $b(\mathcal{C}_\mathbf{L})[t_i : t_j]$ with start timestamp t_i and end timestamp t_j: it contains any post $d \in \mathcal{C}_\mathbf{L}$ whose timestamp is between t_i and t_j within this segment. We write $\mathcal{B}(\mathcal{C}_\mathbf{L}) = \bigcup b(\mathcal{C}_\mathbf{L})[t_i : t_j]$ to denote the *set of all bursts in response to* q.

We let b be short for $b(\mathcal{C}_\mathbf{L})[t_i : t_j]$ in the following. As it does not access the contents of posts, the source of complexity in our burst detection method is in the problem of finding all maximal segments: this problem can be solved in linear time (Ruzzo and Tompa, 1999), so that the computational complexity of our burst detection method is $O(|\mathcal{C}_\mathbf{L}|)$.

4.1.3 Burst-aware fusion

We turn to the key steps 3 and 4 of Algorithm 1 and define our burst-aware fusion algorithm. Motivated by the fact that people tend to talk about a topic within specific short time intervals (Lappas et al., 2009; Mathioudakis et al., 2010; Peetz et al., 2012), we devise a method that allows posts in the same burst to boost the scores of other posts such that posts that are ranked low in a small number of lists can be promoted by posts in the same burst that are ranked highly in many lists. After detecting a set of bursts, we integrate burst information with scores of posts in the lists, scores that were generated by a standard fusion method X, to estimate $P(d|q)$—the final probability that d ($\in \mathcal{C}_\mathbf{L}$) is relevant to query q.

The Model

We use a set of bursts $\mathcal{B}(\mathcal{C}_\mathbf{L})$ as proxies for d in estimating its relevance in response to q. Specifically, we can rewrite the probability of a post d being about q, $P(d|q)$, as:

$$P(d|q) = \sum_{b \in \mathcal{B}(\mathcal{C}_\mathbf{L})} p(d|b, q) \cdot p(b|q), \tag{4.2}$$

where the probability, $p(b|q)$, indicates how likely a set of posts in b produced within the same time interval are relevant to q, and $p(d|b, q)$ indicates how likely d is talking about

[4]The score of a subsequence is the sum of the burst-time scores of the elements in the subsequence.

q and belongs to b. To estimate $p(d|b,q)$, a linear mixture governed by a free parameter μ is used (Kurland and Lee, 2004; Markovits et al., 2012) such that:

$$p_\theta(d|b,q) := (1-\mu)\cdot p(d|q) + \mu \cdot p(d|b), \tag{4.3}$$

where $p(d|q)$ measures the relevance of d to q and $p(d|b)$ indicates how likely d belongs to b. We substitute (4.3) into (4.2), and define our ***BurstFuseX*** model as:

$$\begin{aligned} f_{BurstFuseX}(d;q) &:= \sum_{b\in\mathcal{B}(\mathcal{C}_\mathbf{L})} \{(1-\mu)\cdot p(d|q) + \mu\cdot p(d|b)\}\cdot p(b|q) \\ &= (1-\mu)\cdot p(d|q) + \mu \cdot \sum_{b\in\mathcal{B}(\mathcal{C}_\mathbf{L})} p(d|b)\cdot p(b|q). \end{aligned} \tag{4.4}$$

That is, to obtain a score for post d in response to q, BurstFuseX uses three types of probability. If posts in a burst are talking about q, b will be rewarded as $p(b|q)$ indicates. If d is strongly associated with b, then as indicated by $p(d|b)$, d will be rewarded. Finally, if each burst b in $\mathcal{B}(\mathcal{C}_\mathbf{L})$ talks about q, d itself discusses q and is strongly associated with a burst, then d will be ranked high in the final fused list.

Notice how BurstFuseX can boost the score of posts: if post d ranks low in a single list (i.e., $p(d|q)$ is small) but is contained in a burst, as indicated by $p(d|b)$, then the final fused score of d, $f_{BurstFuseX}(d;q)$, may still be relatively high, which may boost the final ranking of d.

Estimating the Key Components

Our next task is to derive estimates for the following key components in (4.4):

- $p(d|q)$: post-level relevance—how likely d is talking about q.
- $p(b|q)$: burst-level relevance—how likely a set of posts as a whole is talking about q.
- $p(d|b)$: post-burst association strength—how likely d belongs to b.

Post-level relevance. To obtain $p(d|q)$ in (4.4), we apply Bayes' Theorem, such that $p(d|q) = \frac{p(q|d)p(d)}{p(q)}$, where we let $p(q) \propto \sum_{d'\in\mathcal{C}_\mathbf{L}} p(q|d')p(d')$. A uniform prior distribution is assumed for each post $d' \in \mathcal{C}_\mathbf{L}$. So $p(d|q)$ can be rewritten as:

$$p(d|q) \propto \frac{p(q|d)}{\sum_{d'\in\mathcal{C}_\mathbf{L}} p(q|d')}.$$

We use an estimate $p_\theta(q|d) \propto f_X(d;q)$ (Khalaman and Kurland, 2012), where $f_X(d;q)$ is the score of a standard fusion method X for d given q:

$$p_\theta(d|q) := \frac{f_X(d;q)}{\sum_{d'\in\mathcal{C}_\mathbf{L}} f_X(d';q)}, \tag{4.5}$$

which is the normalized standard fusion score reflecting d's relevance to q. Notice that our burst-aware fusion model will reduce to the standard fusion method X if we let $\mu = 0$

in (4.4), as $f_{BurstFuseX}(d;q) = (1-\mu)\cdot p(d|q) \propto f_X(d|q)$ in (4.4) in this case. In other words, the effect of merging result lists according to $f_{BurstFuseX}(d;q)$ will then be the same as that of merging result lists according to $F_X(d;q)$.

Burst-level relevance. Next, to obtain $p(b|q)$ in (4.4), we apply Bayes' Theorem again, such that $p(b|q) = \frac{p(q|b)p(b)}{p(q)}$, where we use the probability rule, and have $p(q) \propto \sum_{b' \in \mathcal{B}(\mathcal{C}_\mathbf{L})} p(q|b')p(b')$. Assuming a uniform prior for each burst in $\mathcal{C}_\mathbf{L}$ for a given q, the probability that a burst b contains information pertaining to q can be represented as:

$$p(b|q) \propto \frac{p(q|b)}{\sum_{b' \in \mathcal{B}(\mathcal{C}_\mathbf{L})} p(q|b')}.$$

A burst may contain posts that have multiple appearances in the lists to be fused. Prior work on representing sets of posts has shown that product-based representations somewhat outperform sum-based representations (Khalaman and Kurland, 2012; Liu and Croft, 2008; Seo and Croft, 2010). Accordingly, we let:

$$p_\theta(q|b) = \prod_{d \in b} p(q|d)^{\frac{1}{|b|}}.$$

As we use an estimate $p_\theta(q|d) \propto f_X(d;q)$ (see above), $p(b|q)$ can be estimated as:

$$p_\theta(b|q) := \frac{\prod_{d \in b} f_X(d;q)^{\frac{1}{|b|}}}{\sum_{b' \in \mathcal{B}(\mathcal{C}_\mathbf{L})} \prod_{d' \in b'} f_X(d';q)^{\frac{1}{|b'|}}}, \tag{4.6}$$

where $|b|$ and $|b'|$ are the number of posts in b and b', respectively.

Post-burst association strength. To obtain $p(d|b)$ in (4.4), we apply Bayes' Theorem again, such that $p(d|b) = \frac{p(b|d)p(d)}{p(b)}$. We observe that $p(b) \propto \sum_{d' \in \mathcal{C}_\mathbf{L}} p(b|d')p(d')$ and assume a uniform prior for the probability of a post, so that $p(d|b)$ can be represented as:

$$p(d|b) \propto \frac{p(b|d)}{\sum_{d' \in \mathcal{C}_\mathbf{L}} p(b|d')}.$$

Here, $p(b|d)$ is the probability of d belonging to b.

Next, we need to estimate $p(b|d)$. Again, we use the product of scores of posts in a burst with the index of $1/|b|$ rather than the average of the sum of the score.[5] We set:

$$p_\theta(b|d) := \prod_{d'' \in b} p(d''|d)^{\frac{1}{|b|}}, \tag{4.7}$$

to estimate $p(b|d)$, where $d'' \in \mathcal{C}_\mathbf{L}$ is a post in b.

Three factors affect the association strength between d and b: the temporal relationship between d and posts $d'' \in b$, the relevance of d given q, and the relevance of d'' given q. We estimate the time relationship between d and d'' as:

$$p_t(d'', d) = \exp\left\{-\frac{(t_{d''} - t_d)^2}{2\sigma_b^2}\right\}.$$

[5] Experimental results show that using products is not statistically significantly different from using sums with a two-tailed paired t-test at a 95% confidence level.

Here, t_d and $t_{d''}$ are the timestamps of post d and d'', respectively, σ_b is the standard deviation of the timestamps in b:

$$\begin{aligned}
\sigma_b^2 &= \frac{\sum_{k=1}^{n_b}\left\{k-\frac{n_b+1}{2}\right\}^2}{n_b} \\
&= \frac{\sum_{k=1}^{n_b}\left\{k^2-k(n_b+1)+\frac{n_b^2+2n_b+1}{4}\right\}}{n_b} \\
&= \frac{\frac{n_b^3+2n_b^2+n_b}{4}+\sum_{k=1}^{n_b}k^2-(n_b+1)\sum_{k=1}^{n_b}k}{n_b} \\
&= \frac{\frac{n_b^3+2n_b^2+n_b}{4}+\frac{n_b(n_b+1)(2n_b+1)}{6}-(n_b+1)\frac{(n_b+1)n_b}{2}}{n_b} \\
&= \frac{n_b^2-1}{12},
\end{aligned} \tag{4.8}$$

where $n_b = j - i + 1$ is the number of different timestamps of posts in the burst b.[6] If $j = i$, we let $\sigma_b = 0.5$ to avoid $\sigma_b = 0$. The bigger the temporal distance between t_d and $t_{d''}$ is, the smaller $p_t(d'', d)$ will be, which means that compared to other posts in burst b, d is rewarded less by post d''.

Now, to estimate $p(d''|d)$ ((4.7)) we build on the following intuition. If d'' is ranked highly, based on a relatively large value of $p(d''|q)$, and d'' and d are produced at almost the same point in time, based on a relatively high value of $p_t(d'', d)$, then d'' should be able to boost d's score. Hence, we estimate $p(d''|d)$ by putting $p_\theta(d''|d) := p(d''|q) \cdot p_t(d'', d) := p_\theta(d''|q) \cdot p_t(d'', d)$. When we substitute this term in (4.7) we obtain:

$$p_\theta(b|d) := \prod_{d'' \in b} \left\{p_\theta(d''|q) \cdot p_t(d'', d)\right\}^{\frac{1}{|b|}}.$$

Putting everything together, we can now estimate the post-burst association strength, $p(d|b)$, as:

$$p_\theta(d|b) = \frac{\prod_{d'' \in b}\left\{p_\theta(d''|q) \cdot p_t(d'', d)\right\}^{\frac{1}{|b|}}}{\sum_{d' \in \mathcal{C}_\mathbf{L}} \prod_{d'' \in b}\left\{p_\theta(d''|q) \cdot p_t(d'', d')\right\}^{\frac{1}{|b|}}}. \tag{4.9}$$

According to (4.9), if d is in b and the scores of posts surrounding d (including d itself) in b are high, the association strength between d and b increases. In this case, d's score will be boosted. Note, by the way, that d ($\in \mathcal{C}_\mathbf{L}$) does not have to be in b; any d in $\mathcal{C}_\mathbf{L}$ can have a non-zero association strength to any b in $\mathcal{B}(\mathcal{C}_\mathbf{L})$.

4.2 Experimental Setup

In this section, we describe our experimental setup, and Section 4.2.1 lists our specific research questions, which together help answer the main research question for this chapter. Finally, Section 4.2.3 and Section 4.2.4 detail how BurstFuseX is trained and optimized, and the settings of the experiments, respectively.

[6] Alternative definitions of σ_b are possible, but we found that this has little effect on overall retrieval performance.

4.2.1 Detailed research questions

We divide our main research question (RQ 1) into the following detailed research questions, and let these questions guide the remainder of the chapter:

RQ 1.1 Does BurstFuseX outperform the standard data fusion method that it integrates? (See Section 4.3.1 for the answer.)

RQ 1.2 Does BurstFuseλ-Merge outperform BurstFuseCombSUM or BurstFuseCombMNZ? (See Section 4.3.1 for the answer.)

RQ 1.3 Does BurstFuseX outperform the best run to be fused? (See Section 4.3.1 for the answer.)

RQ 1.4 What is the effect of using burst information in BurstFuseX? I.e., what is the impact of the free parameter μ in Eq. 4.4? (See Section 4.3.2 for the answer.)

RQ 1.5 What is the effect of the number of lists to be fused in BurstFuseX? (See Section 4.3.3 for the answer.)

RQ 1.6 Can we observe the hypothesized effect sketched in Fig. 4.1 (See Section 4.3.4 for the answer.)

RQ 1.7 How fast is BurstFuseX compared to other data fusion methods? (See Section 4.3.5 for the answer.)

RQ 1.8 Can BurstFuseX beat burst or time-sensitive microblog search algorithms? (See Section 4.3.6 and Section 4.3.7 for the answer.)

RQ 1.9 Can BurstFuseX aid a single run that does not take time into account? (See Section 4.3.8)

4.2.2 Baselines

We compare BurstFuseX to 3 data fusion baselines: 2 traditional unsupervised methods, i.e., CombSectionUM, CombMNZ, and a start-of-the-art supervised method, λ-Merge (Sheldon et al., 2011). As BurstFuseX utilizes burst information (temporal information) to boost the performance, we also compare BurstFuseX to 4 state-of-the-art burst-sensitive microblog search algorithms: time-based language model (TBLM) (Li and Croft, 2003), textual quality factor model with temporal query expansion (LM-T(qe)) (Massoudi et al., 2011), direct time-sensitive BM25 retrieval model (DIRECT-BM25 (mean)) (Dakka et al., 2012) and temporal tweet selection feedback method (TSectionF+QDRM) (Miyanishi et al., 2013b). All of these burst-sensitive algorithms first detect bursts (or time-spans) based on the content (words) of the posts and then utilize the burst information to boost the retrieval performance. Our BurstFuseX detect bursts based on the fusion scores of posts rather than directly based on the content of the posts.

To illustrate the merits of detecting bursts from fusion scores, we implement an alternative algorithm, BurstFuseX$_{posts}$ (BurstFuseCombSUM$_{posts}$ and BurstFuseCombMNZ$_{posts}$), which detects bursts using the burst detection approach presented in (Lappas et al., 2009), using the content of posts, and then fuses the input rank lists using our fusion framework. In other words, the only difference between BurstFuseX and BurstFuseX$_{posts}$ is in the way they detect bursts. To build the index of the dataset that some of our baselines require, we apply Porter stemming, tokenization, and stopword

Table 4.2: Summary of sampled runs from the TREC 2011 Microblog track.

Class	Sampled runs	Performance
Class 1	clarity1, waterlooa3, FASectionILKOM02, isiFDL, DFReeKLIM30, PRISrun1	$0.40 \leq$ p@30
Class 2	KAUSTRerank, ciirRun1, gut, dutirMixFb, normal, UDMicroIDF	$0.30 \leq$ p@30 < 0.40
Class 3	WESTfilext, LThresh, qRefLThresh, run3a, Nestor, uogTrLqea	$0.20 \leq$ p@30 < 0.30
Class 4	FASILKOM02, waterlooa3, gut, UDMicroIDF, run3a, qRefLThresh	$0.20 \leq$ p@30

removal (using INQUERY lists) to posts using the Lemur toolkit.[7] The features and settings used for λ-Merge are briefly described in Appendix 4.A.

4.2.3 Training and optimization

Our BurstFuseX fusion method incorporates a single free parameter, μ in (4.4). The value of μ ($\in \{0, 0.1, \ldots, 1\}$) is set using 10-fold cross validation performed over the entire set of queries in the TREC 2011 Microblog track. In the learning phase, the performance of BurstFuseX is optimized with respect to MAP. In other words, the set of 49 queries is randomly partitioned into 10 equal size subsamples; the performance for a single test subsample (5 queries) is that attained using a value of μ that maximizes MAP performance over the remaining subsamples (44 queries). We repeat the experiment 10 times and report the average scores on the metrics. In each time, the subsamples are permuted until all the 49 queries were chosen once for the test set. The setting of BurstFuseX over the entire set of queries in the TREC 2012 Microblog track is the same as that in TREC 2011 Microblog track. Our baseline fusion methods, i.e., CombSUM, CombMNZ and λ-Merge, do not incorporate free parameters.

4.2.4 Experiments

We report on 8 main experiments in this chapter. The experiments are carried out on the microblog search collection; details of the microblog search collection can be found in Section 3.2.2. First, to understand the overall performance of BurstFuseX, we sample about 10% from the ranked lists produced by the participants in the TREC 2011 and 2012 Microblog tracks based on the lists' p@30 distribution, respectively: 18 out of the 174 runs in TREC 2011 and 18 out of the 117 runs in TREC 2012, 6 each with p@30 scores between 0.20 and 0.30 (Class 3), between 0.30 and 0.40 (Class 2), and over 0.40 (Class 1). We also randomly choose two runs from each class to construct Class 4. See Tables 4.2 and 4.3 for details of our sample runs from the TREC 2011 and 2012 Microblog tracks, respectively. Note that in our experiments, the runs in Class 1 in Tables 4.2 and 4.3 are actually the six best ones in the TREC 2011 and 2012 Microblog tracks, respectively. In every class, we use run1, run2, run3, run4, run5 and run6 to refer to the runs in descending order of p@30 score.

To understand the influence of bursts and see whether burst information is helpful to boost fusion performance and to which extent, we change the parameter μ in Eq. 4.4 from 0.0 to 1.0, which alters the degree to which burst-based and standard fusion information are to be mixed. Then, to understand the effect of the number of lists to be merged,

[7] `http://www.lemurproject.org`

Table 4.3: Summary of sampled runs from the TREC 2012 Microblog track.

Class	Sectionampled runs	Performance
Class 1	hitURLrun3, kobeMHC2, kobeMHC, uwatgclrman, hitQryFBrun4, kobeL2R	$0.40 \leq$ p@30
Class 2	QEWebFB, indri, KLIMLL, UNCRQE, gucasGenReg, KLIMLPLL	$0.30 \leq$ p@30 < 0.40
Class 3	FASectionILKOM01, IIEIR03, RUN2, expansion, IRSIISI, uwatgclrbase	$0.20 \leq$ p@30 < 0.30
Class 4	hitQryFBrun4, kobeL2R, KLIMLL, UNCRQE, IIEIR03, IRSIISI	$0.20 \leq$ p@30

we randomly choose k ($\in \{2, 4, 6, \ldots, 38\}$) lists from the 174 TREC 2011 Microblog lists and fuse them by BurstFuseX and the standard fusion methods. We repeat the experiments 20 times and report the average results as well as the corresponding standard deviation scores.

In order to understand the topic-level performance of BurstFuseX, we provide an analysis of topic-level performance against the standard fusion method it cooperates. Next, to determine how fast BurstFuseX can merge result lists, we again fuse k ($\in \{2, 4, \ldots, 30\}$) lists, and report and compare the average computing time required by BurstFuseX against that of the standard fusion methods. To understand whether BurstFuseX can improve over microblog search approaches that already incorporate time-sensitive search algorithms, we compare the performance of BurstFuseX, the standard fusion method it builds on and 5 time-sensitive baselines of searching microblogs.

To understand whether detecting bursts based on standard fusion scores works better than detecting based on the textual content of posts, we make a comparison between our fusion methods and those detecting bursts based on the textual contents of posts. We also compare burst-sensitive component lists to be fused and the fusion methods to see whether fusion can help to boost the retrieval performance. Finally, to understand whether BurstFuseX requires multiple result lists or if it can aid single runs that may not have taken time into account, we fuse only a single result list using BurstFuseX and compare the single result list and the output of BurstFuseX on that list.

As described in Section 4.1, in our experiments we use two unsupervised data fusion methods, CombSUM and CombMNZ, and one supervised method, λ-Merge, as representatives of the standard methods that can be integrated by BurstFuseX.

4.3 Results and Analysis

In this section, we present our experimental results and perform an analysis. We follow the order of the research questions listed in Section 4.2.1. Particularly, in Section 4.3.1 we examine the effectiveness of BurstFuseX on fusing the sample lists; in Section 4.3.2 we study the effect of using burst information and in Section 4.3.3 the effect of the number of lists on the overall performance; Section 4.3.4 reports on a topic level analysis; Section 4.3.5 is devoted to look at the runtime performance of BurstFuseX and in Section 4.3.6 we examine whether BurstFuseX is able to add anything in terms of performance on top of result lists produced by retrieval methods that already use temporal information; Section 4.3.7 provides a further analysis of the use of burst information in data fusion for microblog search; finally, Section 4.3.8 shows the performance of BurstFuseX on single result list.

4.3.1 Fusing the sample lists

The performance of BurstFuseX and of the standard fusion methods X that it incorporates is detailed in Table 4.4, with scores based on the 10% sample runs from the TREC 2011 Microblog track, as mentioned in Section 4.2.4. It is clear from Table 4.4 that the performance of unsupervised data fusion and the corresponding burst-aware fusion methods, i.e., CombSUM, CombMNZ, BurstFuseCombSUM and BurstFuseCombMNZ, is better than that of the best result list that is used in the merging process (run1) for all classes and on almost all metrics. Many of these improvements are statistically significant. More importantly, in class 1 all of these methods beat the best recorded run (isiFDL) in the TREC 2011 Microblog track (e.g., the p@30 score for BurstFuseCombSUM is 0.5578 while that of the best run in the track is 0.4551), and even the standard fusion method it integrates, i.e., CombSUM, outperforms the best recorded run. Meanwhile, in class 1, the supervised data fusion methods λ-Merge and BurstFuseλ-Merge can also beat the best recorded run in terms of the official metric p@30. All of these demonstrate that data fusion strategies can help improve effectiveness in searching microblogs.

It is worth noting that in most cases BurstFuseX outperforms the standard fusion method X that it incorporates for all classes and on nearly all metrics (MAP, p@10, p@15, p@30). Almost all of these improvements are substantial and statistically significant. For instance, when fusing the runs in class 4, the MAP and p@30 metrics of BurstFuseCombMNZ are 0.2883 and 0.4387, respectively, compared to only 0.2794 and 0.4048, respectively, for CombMNZ. This finding attests to the merits of incorporating burst information into data fusion and shows that using burst information can improve the performance of existing data fusion methods in terms of MAP and p@30.

Interestingly, in Table 4.4 when we consider the p@5 scores, we see that BurstFuseX always outperforms the best single run but that it loses the performance when against the standard fusion method X on which it builds in most cases. One reason is that some relevant posts ranked very high in the lists being merged but not near any of the bursts are forced down the ranking. Another reason is that a very small number of irrelevant posts in the bursts are promoted to slightly higher ranks in the fused list. In contrast, the gain obtained from relevant posts that are ranked at the bottom of the runs but near bursts are clearly observed at bigger cut-offs, resulting in the improvements of p@10, p@15 and p@30 scores.

Additionally, from Table 4.4 we see that, in terms of MAP, BurstFuseCombSUM outperforms BurstFuseCombMNZ, and both of them outperform BurstFuseλ-Merge in Class 2 (0.2651, 0.2587, 0.2161, respectively). In other words, BurstFuseCombSUM outperforms BurstFuseCombMNZ, followed by BurstFuseλ-Merge. This is quite obvious in Class 1, Class 2 and Class 4 for instance. In terms of the standard fusion methods, CombSUM performs almost the same as CombMNZ with no statistically significant differences. Both CombSUM and CombMNZ easily beat the supervised standard fusion method λ-Merge; in same cases, λ-Merge becomes worse than the best result list (run1) to be fused in the corresponding class. This may be due to over-fitting of λ-Merge.

As a sanity check, so as to confirm our observations about the performance of BurstFuseX, we also test BurstFuseX on the 10% sample runs from TREC 2012 Microblog track. We present the experimental results of BurstFuseX and the standard fusion method X it incorporates in Table 4.5. Clearly, our observations about the performance of Burst-

Table 4.4: Retrieval performance on the 10% sample lists from the TREC 2011 Microblog track. Boldface marks the better performance between BurstFustX and the standard fusion method X that it incorporates; a statistically significant difference between the two is marked in the upper right hand corner as ▲ (or ▼) for $\alpha = .01$, or △ (and ▽) for $\alpha = .05$; a statistically significant difference with run1 is marked in the upper left hand corner using the same symbols; the best result per column is underlined.

	Class 1					Class 2				
	MAP	p@5	p@10	p@15	p@30	MAP	p@5	p@10	p@15	p@30
run1	.2210	.5918	.5673	.5347	.4551	.1457	.4612	.4143	.3714	.3571
run2	.2690	.5959	.5796	.5442	.4537	.1886	.4776	.4347	.3878	.3463
run3	.2318	.5755	.5367	.5034	.4401	.1525	.4041	.4143	.3878	.3408
run4	.2058	.5714	.5367	.4939	.4211	.1376	.3959	.3939	.3796	.3218
run5	.2575	.5673	.4980	.4721	.4211	.1688	.3878	.3633	.3605	.3136
run6	.2098	.5469	.5102	.4694	.4095	.1820	.4122	.3796	.3619	.3027
CombSUM	.3404	▲**.6245**	.5816	.5524	▲.4966	▲.2625	▲**.5306**	▲.4531	▲.4286	▲.3735
BurstFuseCombSUM	▲**.3563**▲	▲.6163	▲**.5959**△	▲**.5878**▲	▲**.5578**▲	▲**.2651**▲	△.4898▼	▲**.4694**▲	▲**.4553**▲	▲**.4344**▲
CombMNZ	▲.3385	▲.6245	.5755	.5524	▲.5020	▲.2581	▲**.5347**	▲.4592	▲.4354	▲.3789
BurstFuseCombMNZ	▲**.3528**▲	▲**.6286**	▲**.5959**▲	▲**.5918**▲	▲**.5517**▲	▲**.2587**▲	▲.5061▼	▲**.4735**▲	▲**.4567**△	▲**.4242**▲
λ-Merge	.2548	▼.5641	▼.5631	.5496	.4611	.1898	▽**.4641**	▲.4608	▲.4307	▲.3668
BurstFuseλ-Merge	▲**.2920**▲	▼**.5655**	**.5812**▲	▲**.5701**▲	▲**.5011**▲	▲**.2161**▲	▼.4384▼	▲**.4651**	▲**.4558**▲	▲**.4195**▲

	Class 3					Class 4				
	MAP	p@5	p@10	p@15	p@30	MAP	p@5	p@10	p@15	p@30
run1	.1661	.4041	.3408	.2898	.2122	.2058	.5714	.5367	.4939	.4211
run2	.0997	.3429	.3000	.2653	.2095	.2098	.5469	.5102	.4694	.4095
run3	.1636	.3959	.3122	.2571	.2041	.1376	.3959	.3939	.3796	.3218
run4	.0753	.3265	.2735	.2585	.2034	.1820	.4122	.3796	.3619	.3027
run5	.0571	.2980	.2551	.2408	.2020	.1636	.3959	.3122	.2571	.2041
run6	.0994	.3510	.2735	.2408	.2016	.0753	.3265	.2735	.2585	.2034
CombSUM	▲.2150	▲**.4857**	▲**.4327**	▲.3837	▲.2952	.2795	▲**.6122**	▲.5327	▲.4721	▼.3918
BurstFuseCombSUM	▲**.2283**	▲.4408▼	▲.4184▽	▲**.3973**△	▲**.3388**▲	▲**.2863**▲	▲.5633▼	▲**.5449**△	▲**.5011**▲	▲**.4380**▲
CombMNZ	▲.2187	▲**.4898**	▲**.4327**	▲.3932	▲.2973	.2794	▲**.6000**	▲.5449	△.4830	.4048
BurstFuseCombMNZ	▲**.2313**△	▲.4531▼	▲**.4327**	▲**.4122**	▲**.3442**▲	▲**.2883**▲	▲.5796▼	▲**.5469**	△**.5043**▲	**.4387**▲
λ-Merge	▼.1450	▼**.3757**	▲.3709	▲.3431	▲.2801	▽.1950	▼**.4816**	▼.4708	▲.4628	.4038
BurstFuseλ-Merge	▼**.1513**▲	▼.3547▽	▲**.3810**△	▲**.3572**▲	▲**.3217**▲	▲**.2212**▲	▼.4609▼	▼**.4811**△	▲**.4937**▲	▲**.4303**▲

FuseX and the standard fusion methods when fusing the runs from TREC 2011 Microblog track are confirmed by the 2012 data. For instance, in Table 4.5 we see that all data fusion methods, both BurstFuseX and other ones, when fusing runs in Class 1, outperform the best result run of the TREC 2012 Microblog track, and almost all of these improvements are statistically significant in terms of the metrics listed in the table. Below, we only report experimental results for the TREC 2011 Microblog track test collection: the 2012 collection consistently yields the same overall results and trends.

4.3.2 The use of burst information

Next we examine the effect of using different amounts of burst information in our burst-aware fusion method. Put differently, we examine the impact of the free parameter μ in

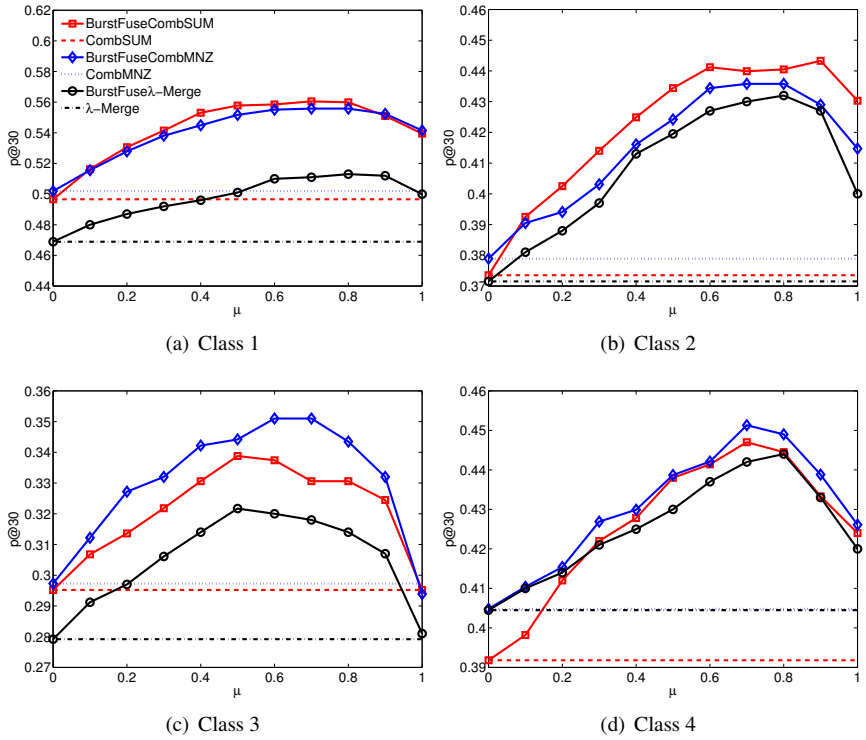
Table 4.5: Retrieval performance on the 10% sample lists from the TREC 2012 Microblog track. Boldface marks the better performance between BurstFustX and the standard fusion method X that it incorporates; a statistically significant difference between the two is marked in the upper right hand corner as $^{\blacktriangle}$ (or $^{\blacktriangledown}$) for $\alpha = .01$, or $^{\triangle}$ (and $^{\triangledown}$) for $\alpha = .05$; a statistically significant difference with run1 is marked in the upper left hand corner using the same symbols; the best result per column is underlined.

	Class 1					Class 2				
	MAP	p@5	p@10	p@15	p@30	MAP	p@5	p@10	p@15	p@30
run1	.1685	.6102	.5729	.5254	.4695	.1136	.4881	.4492	.4271	.3638
run2	.1582	.5898	.5627	.5424	.4684	.1095	.4576	.4017	.3842	.3463
run3	.1566	.5831	.5390	.4949	.4435	.1079	.4136	.3966	.3706	.3390
run4	.1563	.6000	.5763	.5480	.4571	.1078	.4508	.4475	.4305	.3655
run5	.1526	.5729	.5407	.5322	.4610	.1010	.4712	.4322	.4023	.3599
run6	.1385	.5525	.5237	.5028	.4429	.0744	.3898	.3678	.3605	.3209
CombSUM	.1785	$^{\blacktriangle}$**.6271**	.5814	$^{\blacktriangle}$.5559	$^{\blacktriangle}$.5028	$^{\blacktriangle}$.1263	$^{\blacktriangle}$**.5322**	$^{\blacktriangle}$.4898	$^{\blacktriangle}$.4520	$^{\triangle}$.3797
BurstFuseCombSUM	$^{\blacktriangle}$**.2028**$^{\blacktriangle}$	$^{\blacktriangledown}$.5797$^{\blacktriangledown}$	**.5763**	$^{\blacktriangle}$**.5842**$^{\blacktriangle}$	$^{\blacktriangle}$**.5797**$^{\blacktriangle}$	$^{\blacktriangle}$**.1473**$^{\blacktriangle}$	$^{\blacktriangle}$.4983$^{\blacktriangledown}$	$^{\blacktriangle}$**.4898**	$^{\blacktriangle}$**.4734**$^{\blacktriangle}$	$^{\blacktriangle}$**.4548**$^{\blacktriangle}$
CombMNZ	.1776	$^{\blacktriangle}$**.6339**	.5831	$^{\blacktriangle}$.5582	$^{\blacktriangle}$.5011	$^{\blacktriangle}$.1319	$^{\blacktriangle}$**.5492**	$^{\blacktriangle}$.5085	$^{\blacktriangle}$.4610	$^{\blacktriangle}$.3944
BurstFuseCombMNZ	$^{\blacktriangle}$**.2019**$^{\blacktriangle}$	.6000$^{\blacktriangledown}$	**.5847**	$^{\blacktriangle}$**.5887**$^{\blacktriangle}$	$^{\blacktriangle}$**.5678**$^{\blacktriangle}$	$^{\blacktriangle}$**.1516**$^{\blacktriangle}$	$^{\blacktriangle}$.5186$^{\blacktriangledown}$	$^{\blacktriangle}$**.4949**	$^{\blacktriangle}$**.4859**$^{\blacktriangle}$	$^{\blacktriangle}$**.4651**$^{\blacktriangle}$
λ-Merge	.1700	.6134	.5673	$^{\blacktriangle}$.5462	.4718	.1131	$^{\blacktriangle}$**.5124**	$^{\triangle}$.4583	.4367	.3698
BurstFuseλ-Merge	$^{\blacktriangle}$**.1843**$^{\triangle}$	$^{\blacktriangledown}$**.5734**$^{\blacktriangledown}$	**.5730**	$^{\blacktriangle}$**.5675**$^{\blacktriangle}$	$^{\blacktriangle}$**.5184**$^{\blacktriangle}$	$^{\blacktriangle}$**.1326**$^{\blacktriangle}$	$^{\blacktriangle}$.4878$^{\blacktriangledown}$	$^{\blacktriangle}$**.4766**$^{\blacktriangle}$	$^{\blacktriangle}$**.4638**$^{\blacktriangle}$	$^{\blacktriangle}$**.4281**$^{\blacktriangle}$

	Class 3					Class 4				
	MAP	p@5	p@10	p@15	p@30	MAP	p@5	p@10	p@15	p@30
run1	.0724	.3390	.3390	.3073	.2542	.1566	.5831	.5390	.4949	.4435
run2	.0694	.3322	.3102	.2983	.2712	.1385	.5525	.5237	.5028	.4429
run3	.0661	.3356	.3390	.3062	.2678	.1095	.4576	.4017	.3842	.3463
run4	.0658	.3017	.2864	.2938	.2480	.1010	.4712	.4322	.4023	.3599
run5	.0626	.2949	.2661	.2520	.2282	.0661	.3356	.3390	.3062	.2678
run6	.0343	.1831	.2068	.2147	.2367	.0343	.1831	.2068	.2147	.2367
CombSUM	$^{\blacktriangle}$.1082	$^{\blacktriangle}$**.5051**	$^{\blacktriangle}$**.4576**	$^{\blacktriangle}$.4158	$^{\blacktriangle}$.3249	$^{\blacktriangledown}$.1360	$^{\blacktriangledown}$**.5559**	$^{\triangledown}$.5288	.4949	$^{\blacktriangledown}$.3955
BurstFuseCombSUM	$^{\blacktriangle}$**.1228**$^{\blacktriangle}$	$^{\blacktriangle}$.4508$^{\blacktriangledown}$	$^{\blacktriangle}$.4424$^{\triangledown}$	$^{\blacktriangle}$**.4260**$^{\triangle}$	$^{\blacktriangle}$**.3904**$^{\blacktriangle}$	**.1584**$^{\blacktriangle}$	$^{\blacktriangledown}$.5220$^{\blacktriangledown}$	$^{\blacktriangledown}$**.5102**$^{\triangledown}$	**.4938**	$^{\blacktriangle}$**.4774**$^{\blacktriangle}$
CombMNZ	$^{\blacktriangle}$.1163	$^{\blacktriangle}$**.5186**	$^{\blacktriangle}$.4746	$^{\blacktriangle}$.4362	$^{\blacktriangle}$.3508	$^{\triangledown}$.1456	$^{\blacktriangledown}$**.5593**	.5356	.5062	$^{\triangledown}$.4260
BurstFuseCombMNZ	$^{\blacktriangle}$**.1326**$^{\blacktriangle}$	$^{\blacktriangle}$.4780$^{\blacktriangledown}$	$^{\blacktriangle}$.4797	$^{\blacktriangle}$**.4554**$^{\triangle}$	$^{\blacktriangle}$**.4147**$^{\blacktriangle}$	$^{\triangle}$**.1661**$^{\blacktriangle}$	$^{\blacktriangledown}$.5254$^{\blacktriangledown}$	$^{\blacktriangledown}$**.5300**	$^{\blacktriangle}$**.5153**	$^{\blacktriangle}$**.4842**$^{\blacktriangle}$
λ-Merge	$^{\blacktriangle}$.1075	$^{\blacktriangle}$**.4983**	$^{\blacktriangle}$.4458	$^{\blacktriangle}$.4037	$^{\blacktriangle}$.3352	$^{\blacktriangledown}$.1301	$^{\blacktriangledown}$**.5416**	$^{\blacktriangledown}$.5119	.4863	$^{\blacktriangledown}$.3847
BurstFuseλ-Merge	$^{\blacktriangle}$**.1104**$^{\blacktriangledown}$	$^{\blacktriangle}$.4684$^{\blacktriangledown}$	$^{\blacktriangle}$**.4482**	$^{\blacktriangle}$**.4267**$^{\blacktriangle}$	$^{\blacktriangle}$**.3982**$^{\blacktriangle}$	**.1488**$^{\blacktriangle}$	$^{\blacktriangledown}$.5362	$^{\blacktriangledown}$**.5107**	**.4954**	**.4477**$^{\blacktriangle}$

(4.4). Fig. 4.3 depicts the MAP and p@30 performance curves for BurstFuseX and the corresponding standard fusion methods it integrates when fusing result lists in Class 1, Class 2, Class 3 and Class 4, respectively. For $\mu = 0$, BurstFuseX amounts to the standard fusion method X that it integrates; more weight is put on burst information with higher values of μ; for $0 < \mu < 1$, the standard fusion scores of posts (according to method X) as well as the burst information are utilized for fusing the lists.

It is worth noting in Fig. 4.3 that when fusing lists in different quality classes, the official metric p@30 scores of BurstFuseX ($\mu > 0$) are usually higher than those of the standard fusion method X it incorporates ($\mu = 0$), especially when $\mu = 0.6, 0.7$. For instance, in Class 4 the p@30 performance of BurstFuseCombSUM peaks at $\mu = 0.7$ with the score of 0.4451, while that of the standard fusion method it integrates, CombSUM achieves only 0.3918. As we observed before, BurstFuseX works better when it



(a) Class 1 (b) Class 2

(c) Class 3 (d) Class 4

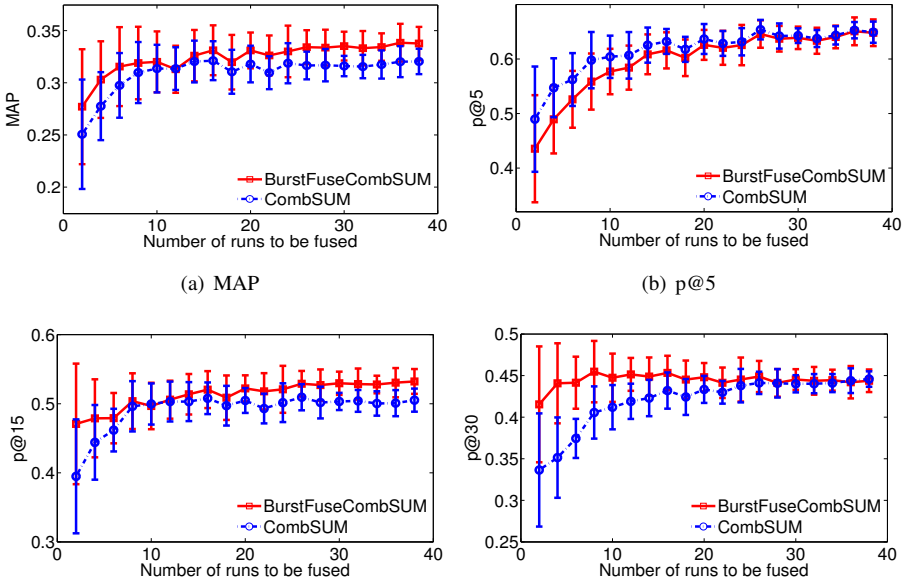
Figure 4.3: Effect of varying the value of μ on the P@30 performance of BurstFuseX when merging lists in (a) Class 1, (b) Class 2, (c) Class 3 and (d) Class 4. When $\mu = 0$, BurstFuseX amounts to the standard fusion method X that it integrates. More weight is put on burst information with higher value of μ. Note: figures are not to the same scale.

integrates one of the unsupervised standard fusion methods than the supervised fusion method λ-Merge.

In addition, it is clear from Fig. 4.3 that when the quality of the result lists as a whole is higher, it will be more useful to utilize burst information. Fig. 4.3 shows that even if $\mu = 1.0$, BurstFuseX still outperforms the standard fusion method it incorporates in high quality classes, like Class 1 and Class 2. But when the quality of the result lists as a whole becomes lower, the positive impact of BurstFuseX is reduced. We provide a further analysis of the use of burst information in data fusion and single retrieval microblog search algorithms in Section 4.3.7.

4.3.3 Effect of the number of lists to be merged

We have already seen that BurstFuseX outperforms the standard fusion methods it incorporates when fusing 6 lists in different quality classes. We now explore the effect on the performance of BurstFuseX of varying the number of lists being merged, in terms of MAP, p@5, p@15 and p@30. In Fig. 4.4, we randomly choose $k \in \{2, 4, 6, \ldots, 38\}$ lists



(a) MAP (b) p@5

(c) p@15 (d) p@30

Figure 4.4: Effect of the number of lists to be merged, k, on (a) MAP (b) p@5 (c) p@15 and (d) p@30. The scores of performance are with the corresponding standard deviation. Note: figures are not to the same scale.

from the 174 lists made available by the TREC 2011 Microblog track and then fuse them. For each k, we repeat the experiment 20 times and report on the average scores as well as the standard deviation. We use CombSUM and BurstFuseCombSUM as a representative example; for the other combinations with a standard fusion method X qualitatively similar results can be observed.

As can be seen in Fig. 4.4, with fewer than 12 lists to be merged by either BurstFuseX or the standard fusion method it builds on, the addition of one more list tends to lead to performance increases across all metrics. Beyond 12 lists, the performance gains of additional lists tend to level off. This is because despite the fact that with more lists, on average we may see more high quality lists, more low quality lists may show up as well. Unlike the MAP, p@15 and p@30 performance of BurstFuseCombSUM where it always enhances that of CombSUM, the early precision p@5 performance of BurstCombSUM is worse than that of CombSUM especially when fewer lists are merged. This observation is consistent with those in both Tables 4.4 and 4.5. Performance gains in terms of p@15 of BurstFuseCombSUM and CombSUM continue even when more than 16 lists are being fused. In contrast, in terms of p@30, when the number of lists to be merged increases, the gain of BurstFuseCombSUM over CombSUM decreases. As more lists are being fused, there is no further gain in terms of burst-awareness.

4.3.4 Topic-level analysis

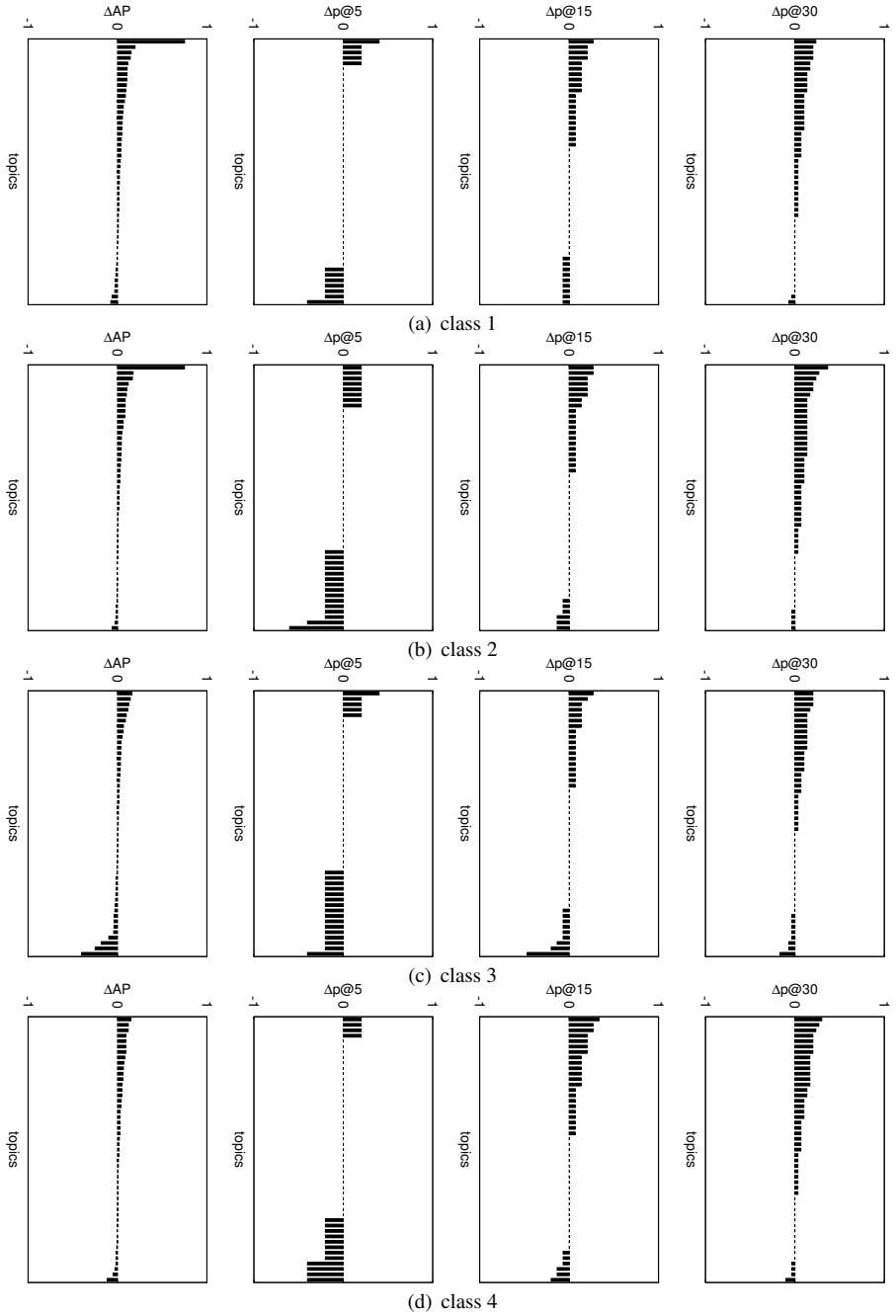
Next, we take a closer look at per query improvements of BurstFuseX over the underlying standard fusion method X. For brevity, we only consider BurstFuseCombSUM as a representative to report all the queries (topics) performance differences against that of the standard fusion method it incorporates, and we only consider runs in Class 1, Class 2, Class 3 and Class 4, respectively. The results for other instances of BurstFuseX are qualitatively similar.

Fig. 4.5 shows the per query performance differences in terms of AP, p@5, p@15 and p@30, respectively, between BurstFuseCombSUM and CombSUM. Overall, gains by BurstFuseCombSUM over CombSUM outnumber losses for p@15 and p@30 as well as MAP, but not for very early precision, i.e., p@5. Gains by BurstFuseCombSUM over CombSUM are due mainly to topics that are discussed only in very specific time intervals. Examples include topics MB010 (Egyptian protesters attack museum), MB011 (Kubica crash) and MB015 (William and Kate fax save-the-date). Invariably, for such topics we found evidence of the intuition depicted in Fig. 4.1: posts that are ranked low in a small number of lists but that are pushed into the final merged list by BurstFuseX because they are central to a burst. For instance, in response to topic MB010 (Egyptian protesters attack museum), post #30354903104749568 is ranked near the bottom in only two lists (at ranks 26 and 27 in runs clarity1 and DFReekLIM30, respectively). Because many posts for the topic were generated around the same time interval (January 26–29, 2011, when the event happened) and are ranked highly in many lists to be fused, post #30354903104749568 is rewarded and ranked as high as top 6 in the merged list.

Topics for which BurstFuseCombSUM cannot beat ComSUM tend to be quite general and unrelated to any specific time windows. Examples include topics MB023 (Amtrak train service) and MB027 (reduce energy consumption). For a very small number of queries, BurstFuseCombSUM's performance, in terms of MAP or p@30 is worse than that of CombSUM. One reason that we observed for this phenomenon is that a very small number of posts are not relevant to the topics even if they are central to the bursts according to their timestamps, and hence they should not be rewarded. An example here is topic MB031 (Special Olympics athletes). In response to this topic, result lists to be fused ranked some irrelevant posts highly, but these posts are still in the bursts, which results in promoting these posts to high ranks, which hurts the performance.

4.3.5 Run-time analysis

We now examine the run-times of BurstFuseX. In particular, we explore what the added costs in terms of run-time of BurstFuseX is on top of the standard fusion methods that it incorporates. Our implementation of BurstFuseX is developed in C++ and the experiments are run on a 10.6.8 MacBook Pro computer with 4GB memory and a 2.3 GHz Intel core i5 processor. In Table 4.6, we randomly choose $k \in \{2, 4, 6, 8, 12, 18, 24, 30\}$ lists from the 174 lists available from the TREC 2011 Microblog track. For each k, we repeat the following experiment 20 times: sample a query from the TREC 2011 Microblog track test set, run fuse k result lists for the query (using CombSUM, CombMNZ, λ-Merge as well as BurstFuseCombSUM, BurstFuseCombMNZ and BurstFuseλ-Merge), record the wall clock time. The results are recorded in Table 4.6 and plotted in Fig. 4.6.



(a) class 1

(b) class 2

(c) class 3

(d) class 4

Figure 4.5: Per topic performance differences of BurstFuseCombSUM against CombSUM. The figures shown are for the 2011 Microblog track sampled runs in Class 1, Class 2, Class 3 and Class 4, for AP, p@5, p@15 and p@30 difference. A bar extending to the right of the center of a plot indicates that BurstFuseCombSUM outperforms CombSUM, and vice versa for bars extending to the left of the center.

Table 4.6: Time spent on fusing lists by different aggregation methods. Recorded in seconds with standard deviations (std).

	Number of lists							
	2	4	6	8	12	18	24	30
CombSUM	3.06e–4	5.01e–4	5.76e–4	9.33e–4	1.03e–3	1.98e–3	2.77e–3	3.37e–3
std	1.13e–5	1.27e–5	2.57e–5	3.61e–5	6.93e–5	6.49e–5	7.02e–5	7.50e–5
CombMNZ	3.06e–4	5.01e–4	5.76e–4	9.33e–4	1.03e–3	1.99e–3	2.79e–3	3.38e–3
std	1.13e–5	1.27e–5	2.57e–5	3.61e–5	6.93e–5	6.52e–5	6.98e–5	7.01e–5
λ-Merge	1.15	2.55	3.82	5.24	7.78	12.03	16.32	20.74
std	1.22e–1	2.3e–1	5.82e–1	6.74e-1	8.03e–1	1.09	1.13	1.18
BurstFuseCombSUM	9.12e–4	1.50e–3	1.84e–3	2.96e–3	3.43e–3	6.69e–3	9.52e–3	1.38e–2
std	3.42e–5	3.94e–5	7.91e–5	7.41e–5	8.72e–5	1.88e–4	2.66e–4	2.84e–4
BurstFuseCombMNZ	9.12e–4	1.50e–3	1.84e–3	2.96e–3	3.43e–3	6.69e–3	9.52e–3	1.38e–2
std	3.42e–5	3.94e–5	7.91e–5	7.41e–5	8.72e–5	1.88e–4	2.66e–4	2.84e–4
BurstFuseλ-Merge	1.16	2.56	3.85	5.25	7.79	12.04	16.34	20.77
std	1.37e–1	2.57e–1	5.11e–1	6.30e-1	7.78e-1	1.10	1.02	2.41

As can be seen in Table 4.6 and Fig. 4.6, the overhead of running BurstFuseX over simply running the standard fusion method X is very limited, but increases with the number of lists to be merged, especially when BurstFuseX incorporates with CombSUM and CombMNZ. BurstFuseCombSUM and BurstFuseCombMNZ merge the lists within 0.01s when given 30 result lists and within 0.001s when fusing two lists. In contrast, however, compared to any of the fusion methods, BurstFuseλ-Merge has to spend more time, which is almost the same as that of the fusion method it builds on. In addition, it is worth noting in Fig. 4.6 that in many cases, as the number of lists to be fused increases, the time spent on fusing is almost linear for BurstFuseX and the standard fusion method as well. For instance, the time needed to fuse 8 lists by BurstFuseCombSUM is nearly double the time needed for fusing 4 lists (2.96e-3s and 1.50e-3s, respectively).

4.3.6 Effect of fusing time-sensitive result lists

BurstFuseX uses temporal information in an essential way. What happens when it fuses result lists that have been generated using temporal information themselves? That is, is there anything left to gain by using BurstFuseX? To answer this question, we explore the performance of BurstFuseX using five result lists that themselves consider temporal information: isiFDRML (Metzler and Cai, 2011; Metzler et al., 2012), DFReeKLIM30 (Horn et al., 2011), Wise2ndRun (Wei et al., 2011), ICTNET11MBR3 (Cao et al., 2011) and UDMicroIDFD (Amati et al., 2011). We use BurstFuseX as well as the standard fusion methods, CombSUM, CombMNZ and λ-Merge to fuse those result lists, and report the comparison results.

Table 4.7 shows the result of the comparisons between BurstFuseX and the five time-sensitive result lists. Obviously, CombSUM and CombMNZ perform on a par with the best result list (isiFDRML). For all metrics but p@5, BurstFuseCombSUM and BurstFuseCombMNZ outperform the best result run as well as the standard fusion method they incorporate; many of the improvements are statistically significant. This illustrates that exploring time information in data fusion has a different effect than utilizing time

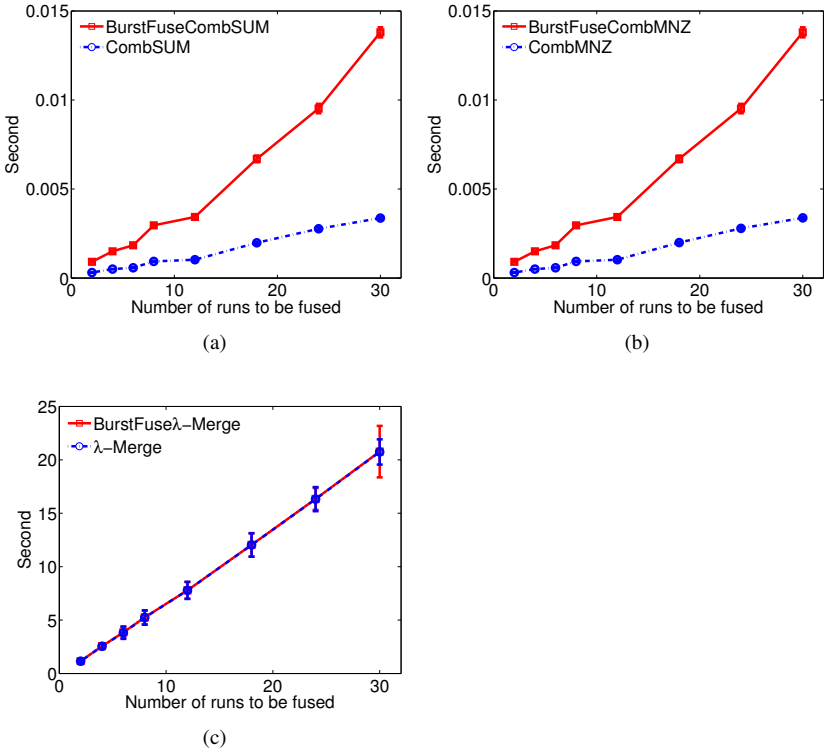


Figure 4.6: Run-times of BurstFuseX against the standard fusion method it builds on with standard deviation. From left to right: (a) BurstFuseCombSUM against CombSUM, (b) BurstFuseCombMNZ against CombMNZ, and (c) BurstFuseλ-Merge against λ-Merge. Note: figures are not to the same scale.

information in an individual ranking function, an effect that can lead to performance increases. One of the main reasons behind this is that posts within intervals in which many relevant posts appear can only be confirmed to be relevant by gathering data from multiple lists, time-sensitive or not. Finally, neither λ-Merge nor BurstFuseλ-Merge can beat the best time-sensitive result list; as before, BurstFuseλ-Merge does improve over λ-Merge.

4.3.7 Further analysis of using burst information

We provide a further analysis of the use of burst information in the setting of microblog search. More specifically, the component lists that we consider next are generated using a time-based language model for microblogs (TBLM) (Li and Croft, 2003), a textual quality factor model with temporal query expansion (LM-T(qe)) (Massoudi et al., 2011), a direct time-sensitive BM25 retrieval model (DIRECT-BM25 (mean)) (Dakka et al., 2012) and a temporal tweet selection feedback method (TSF+QDRM) (Miyanishi et al.,

Table 4.7: Retrieval performance on 5 time-sensitive result lists. Boldface marks the better performance between BurstFustX and the standard fusion method X that it incorporates; a statistically significant difference between the two is marked in the upper right hand corner as $^{\blacktriangle}$ (or $^{\blacktriangledown}$) for $\alpha = .01$, or $^{\triangle}$ (and $^{\triangledown}$) for $\alpha = .05$; a statistically significant difference with the best single run (isiFDRML) is marked in the upper left hand corner using the same symbols; the best result per column is underlined.

	MAP	p@5	p@10	p@15	p@30
isiFDRML	.2326	.6286	.5633	.5374	.4442
DFReeKLIM30	.2318	.5755	.5367	.5034	.4401
Wise2ndRun	.1971	.4980	.4612	.4231	.3639
ICTNET11MBR3	.1863	.4490	.3735	.3619	.3054
UDMicroIDFD	.1428	.3224	.3143	.3075	.2687
CombSUM	.2397	**.6280**	.5673	.5401	.4469
BurstFuseCombSUM	$^{\blacktriangle}$**.3053**$^{\blacktriangle}$	$^{\blacktriangledown}$.5837$^{\blacktriangledown}$	$^{\blacktriangle}$**.5714**	$^{\blacktriangle}$**.5701**$^{\blacktriangle}$	$^{\blacktriangle}$**.4966**$^{\blacktriangle}$
CombMNZ	$^{\triangle}$.2421	**.6284**	.5780	$^{\blacktriangledown}$.5048	$^{\triangle}$.4578
BurstFuseCombMNZ	$^{\blacktriangle}$**.3204**$^{\blacktriangle}$	.6245	$^{\blacktriangle}$**.5878**	$^{\blacktriangle}$**.5755**$^{\blacktriangle}$	$^{\blacktriangle}$**.5116**$^{\blacktriangle}$
λ-Merge	$^{\blacktriangledown}$.2148	$^{\blacktriangledown}$**.5539**	$^{\blacktriangledown}$.5144	$^{\blacktriangledown}$.4873	$^{\blacktriangledown}$.3894
BurstFuseλ-Merge	$^{\blacktriangledown}$**.2271**$^{\blacktriangle}$	$^{\blacktriangledown}$.5501	$^{\blacktriangledown}$**.5257**$^{\triangle}$	$^{\blacktriangledown}$**.5187**$^{\blacktriangle}$	$^{\triangledown}$**.4335**$^{\blacktriangle}$

2013b). To make λ-Merge be comparable, we also add the retrieval scores computed by these 4 time-sensitive retrieval baselines into λ-Merge.

Table 4.8 shows a comparison between fusion methods and the component lists that are burst-sensitive. As can be seen in the table, except for the performance of λ-Merge, which is slightly worse than that of the best component list, all other fusion methods can boost retrieval performance, especially for the fusion methods that we propose in this chapter. This finding again underlines the merit of using fusion for searching microblog posts and of using bursts information in the fusion step.

In Table 4.8 we also compare our BurstFuseX with CombSUM, CombMNZ, λ-Merge, and BurstFuseX$_{posts}$, where BurstFuseX$_{posts}$ is the fusion method that detects bursts based on the content of posts by the detection approached in (Lappas et al., 2009) and then integrates burst information into the fusion process. Clearly, BurstFuseX can still significantly enhance the retrieval performance in all cases and outperforms all standard fusion methods and the best component run, in contrast with BurstFuseX$_{posts}$. We also see that detecting bursts based on one of the standard fusion methods, CombSUM or CombMNZ, works better than detecting bursts based on the content of posts.

4.3.8 Performance on single result list

Finally, to address our final research question, ix, and understand whether BurstFuseX requires multiple result lists or whether it can aid single runs that may not have taken time into consideration, we feed BurstFuseX single result lists and compare the output against the single input list.

Table 4.8: Performance on 4 burst-sensitive result lists. Boldface marks the best result per metric; the best score of component lists per metric is underline; a statistically significant difference between a fusion method and the best component list is marked in the upper left hand corner of the fusion score; a statistically significant difference between BurstFuseX$_{posts}$ and BurstFuseX is marked in the upper right hand corner of the BurstFuseX score.

	MAP	p@5	p@10	p@15	p@30
TSF+QDRM	.2834	.6220	.6856	.6279	.5368
DIRECT-BM25 (mean)	.2798	.6187	.6725	.6320	.5133
LM-T (qe)	.2346	.5836	.5648	.5178	.4471
TBLM	.2231	.5742	.5433	.5017	.4395
CombSUM	△.2962	△.6395	▲.6973	▲.6482	▲.5513
BurstFuseCombSUM$_{posts}$	△.3027	△.6398	▲.6979	▲.6487	▲.5557
BurstFuseCombSUM	△**.3047**	△**.6408**	▲**.6983**	▲**.6497**	▲**.5613**△
CombMNZ	△.2948	△.6350	△.6918	△.6347	△.5420
BurstFuseCombMNZ$_{posts}$	△.3015	△.6373	△.6950	△.6447	△.5433
BurstFuseCombMNZ	△.3027	△.6398	▲.6972	▲.6443	▲.5524▲
λ-Merge	.2847	.6275	.6876	6279	.5383
BurstFuseλ-Merge$_{posts}$	.2881	.6297	.6876	.6313	.5413
BurstFuseλ-Merge	.2942	.6387	.6894	.6326	△.5428

Table 4.9 shows the results on the result lists in class 1; results on other result lists are qualitatively similar. As can be seen in the table, the retrieval performance of BurstFuseX is almost the same as that of the input result list and the difference between them is not statistically significant for any of the metrics. The main reason why BurstFuseX cannot significantly beat the input result list is that detecting bursts within a small set of documents (i.e., those contained in a single result list) is challenging.

4.4 Conclusion

Various features of microblog posts make searching such posts a real challenge: their limited length, their dynamic nature, the creative language usage and their highly contextualized nature. However, the special nature of microblog posts also offers unique opportunities. In this chapter, we have focused on utilizing one such special feature for boosting the performance of search algorithms for microblog posts. We have proposed a data fusion approach, BurstFuseX, that fuses result lists based in part on the bursty nature of many discussions on microblog platforms. Our approach is based on integrating information generated by a standard fusion method, such as CombSUM, CombMNZ or λ-Merge, detecting bursts of posts across the lists being fused, and rewarding posts that are published in or near a burst containing highly ranked posts. Our experiments provide answers to the main research question raised at the beginning of this chapter:

Table 4.9: Performance of BurstFuseX on individual result lists in Class 1. None of the differences between BurstFuseX and the corresponding single input result list is statistically significant.

	MAP	p@5	p@10	p@15	p@30
run1	.2590	.5959	.5796	.5442	.4537
BurstFuse applied to run1	.2593	.5959	.5837	.5442	.4537
run2	.2575	.5673	.4980	.4721	.4211
BurstFuse applied to run2	.2577	.5673	.5000	.4721	.4211
run3	.2318	.5755	.5367	.5034	.4401
BurstFuse applied to run3	.2319	.5755	.5367	.5048	.4401
run4	.2210	.5918	.5673	.5347	.4551
BurstFuse applied to run4	.2210	.5918	.5673	.5347	.4551
run5	.2098	.5469	.5102	.4694	.4095
BurstFuse applied to run5	.2098	.5469	.5102	.4707	.4095
run6	.2058	.5714	.5367	.4939	.4211
BurstFuse applied to run6	.2062	.5755	.5367	.4952	.4211

RQ 1 Can data fusion help microblog search?

To answer the question, we work with the TREC 2011 and 2012 Microblog tracks dataset and fuse the result lists submitted to the TREC using our proposed methods and the baseline methods. Our experimental results show that data fusion can indeed help to improve the performance of searching posts in microblogging environment. The results also show that detecting bursts and then using burst information as part of a standard fusion method can enhance the retrieval performance compared to the standard fusion method it integrates, in terms of mean average precision as well as precision-oriented measures. Our new fusion method has a strong recall-enhancing effect; compared to the standard fusion method it incorporates, this comes at a small price in terms of a small drop in very early precision measures such as p@5. Our experimental results also show that our BurstFuseX method can significantly outperform burst or time-sensitive retrieval models and models that detect bursts based on the content of posts.

As to future work, we have only explored data fusion techniques in microblog search. But data fusion can be, and has been, applied in a variety of areas in IR, like aggregating federated search (Crestani and Markov, 2013; Shokouhi and Si, 2011), cross-lingual search (Si et al., 2008), and finding groups of knowledgeable experts (Liang and de Rijke, 2013). How to apply our burst-aware data fusion in these other areas is an open research question. One avenue for future work is to integrate temporal information into web search—for so-called fresh results. Besides, one of the standard fusion method used in our experiments is λ-Merge, where we assume that labeled data is available. However, this assumption is not always true. Therefore, another promising direction for future work is to explore using pseudo test collections (Asadi et al., 2011; Berendsen et al., 2013) to train a data fusion model so as to enhance the performance.

The proposed data fusion methods in this chapter ignore the rank scores of missing documents (see Fig. 4.1 for what are missing documents). In the next chapter, we will turn to inferring the fusion scores of missing documents while we still integrate temporal information to boost retrieval performance of searching posts in data fusion strategies.

4.A Detailing λ-Merge

In this appendix we detail one of the standard fusion methods we use, λ-Merge. To be able to define the fusion score for document d in response to query q according to λ-Merge, we need to consider the sum of weighting the individual document scores in each list by the weight of the corresponding list:

$$g(d;q) = \sum_m \alpha_m \cdot f(\boldsymbol{x}_d^{L_m};\boldsymbol{\theta}), \tag{4.10}$$

where α_m is the weight of list L_m, $f(\boldsymbol{x}_d^{L_m};\boldsymbol{\theta})$ is the scoring function for d in L_m, with parameters $\boldsymbol{\theta}$. We adapt a linear function for $f(\boldsymbol{x}_d^{L_m};\boldsymbol{\theta})$, due to its widespread use (Atrey et al., 2010), such that:

$$f(\boldsymbol{x}_d^{L_m};\boldsymbol{\theta}) = \sum_n \theta_n \cdot {x_n}_d^{L_m},$$

where θ_n, the n-th dimension of vector $\boldsymbol{\theta}$, is the weight of the n-th feature, and ${x_n}_d^{L_m}$, the n-th dimension of $\boldsymbol{x}_d^{L_m}$, is the value of the n-th feature of d in L_m.

Now, writing C to denote the smoothed objective, according to λ-Merge the parameters α_m and θ_n can be updated based on the gradients $\partial C/\partial\alpha_m = \sum_d(\partial C/\partial g(d;q)) \cdot (\partial g(d;q)/\partial\alpha_m)$ and $\partial C/\partial\theta_n = \sum_d(\partial C/\partial g(d;q))\cdot(\partial g(d;q)/\partial\theta_n)$, respectively. Then, $\partial C/\partial g(d;q)$ is defined as:

$$\partial C/\partial g(d;q) = \sum_e |\Delta_{de}|\{\mathbb{I}_{d\succ e} - 1/(1+\exp(g(e;q)-g(d;q)))\},$$

where $|\Delta_{de}|$ is the absolute change in the performance metric if document d and e were swapped in the current ranking, and the indicator $\mathbb{I}_{d\succ e}$ is 1 when d is judged more relevant than e, 0 otherwise. As explained in the main text, MAP is the metric on which we focus; hence, we optimize λ-Merge for MAP.

Let r_d and r_e denote the rank positions of d and e in L_m. Assume that d and e are misranked by current function g, i.e., $r_d > r_e$ but the relevance level of d, $l(d)$, is larger than that of e, $l(e)$, then

$$|\Delta_{de}| = \frac{1}{R}\left(\sum_{k=r_e}^{r_d} l(k)P@k - \sum_{k=r_e}^{r_d} l'(k)P'@k\right).$$

Here, R is the number of relevant documents for that query, $l(k)$ is the relevance level of the document at rank position k, $P@k$ is the precision at rank k, and $l'(k)$ is the relevance value after the documents at positions r_d and r_e being swapped. Mathematically, the remaining derivatives can be presented as: $\partial g(d;q)/\partial\alpha_m = f(\boldsymbol{x}_d^{L_m};\boldsymbol{\theta})$, and $\partial g(d;q)/\partial\theta_n = \sum_m \alpha_m \cdot {x_n}_d^{L_m}$. After training, the parameters $\boldsymbol{\theta}$ and the weight α_m of each list are obtained. Then we employ max-min normalization to $g(d;q)$, such that the fusion score of d computed by λ-Merge can be obtained as:

$$f_{\lambda\text{-}Merge}(d;q) := \frac{g(d;q)-\min(g_q)}{\max(g_q)-\min(g_q)}, \tag{4.11}$$

Table 4.10: Features used for λ-Merge

Feature	Gloss
Query-post level	
Rank	Inverse of the rank of a post over the number of returned documents
Rankers	Percentage of rankers a post appears in
IsTop-N	If a post is within the top-N results
Post level	
Link	If a post has links
Hashtag	If a post has hashtags
Retweet	If a post has retweets
Density	A post's content quality
Capitalization	A post's textual quality
Length	A post's length deviation from the median length
Post-burstiness	A post's score associated with bursts (The definition of bursts can be found in Section 4.1.2)

where $\min(g_q)$ and $\max(g_q)$ are the minimum and maximum value of g in response to q. For further details about λ-Merge we refer to (Donmez et al., 2008; Sheldon et al., 2011).

Before training λ-Merge, a number of features have to be extracted for $d \in \mathcal{C}_{\mathcal{L}}$. Table 4.10 lists the features used to construct our version of λ-Merge. We identify ten features, extracted from two levels: query-post level and post level; all features are represented by either binary or real numbers. At the query-post level, following Tsai et al. (2008), we use three features: *Rank*, *Rankers* and *IsTop-*N. *Rank* is defined in (2.1). *Rankers* is the number of ranked lists in which the post appears divided by the total number of lists to be merged. *IsTop-*N is a binary feature to indicate if this document is within the top-N results in the list. At the post level, we extract features capable of indicating the quality of a post (Lin et al., 2012; Macdonald et al., 2011); the post features include *Link*, *Hashtag*, *Retweet* to indicate if the document contains links, hashtags, and retweets. The post-level features also consist of content quality indicators of a post (*Density*, *Capitalization* and *Length*) (Lee and et al., 2011; Weerkamp and de Rijke, 2012). We also extract a feature that we call "post-burstiness" based on bursts. *Density* of a post is defined as the sum of $tf \cdot idf$ values of non-stopwords, divided by the number of stopwords they are apart, squared (Lee and et al., 2011). As in (4.11) we use max-min normalization to normalize the scores. The *Capitalization* score of d is obtained by determining the fraction of sentences in d that have a leading capital, seeing to which degree this deviates from the median and then applying max-min normalization (Weerkamp and de Rijke, 2012). The *Length* score of d is obtained by considering the deviation from the median length of posts in the collection. The *Post-burstiness* score of d is obtained by utilizing burst information.

5 Time-Aware Data Fusion

We have introduced a new data fusion approach, BurstFuseX, for microblog search in Chapter 4. As can be seen in Chapter 4, data fusion approaches can inherit the merits of the individual search algorithms whose outputs are being fused. Also, they may boost recall (Wu, 2012) and because they fuse the results of multiple strategies, they mitigate risks associated with opting for a single strategy (Wu, 2012). In BurstFuseX as well as previous work on data fusion, the fusion score of a document is the sum of rank scores from individual input result lists. In addition, they often assume, either implicitly or explicitly, that the rank score of a document is set to zero if the document does not appear in a component result list. In this chapter we challenge this assumption and try to infer the rank scores of missing documents to improve the fusion performance. We are inspired and motivated by the success of matrix factorization in collaborative filtering (CF) (Goldberg et al., 1992) for inferring the rating of unobserved products. Collaborative filtering methods first model users with latent interests and the products with latent features by matrix factorization, and then try to predict the rating of products for the given users with the observations of the existing users' rating data (Kurucz et al., 2007; Ma et al., 2011a; Salakhutdinov and Mnih, 2008a,b). Thus, we are interested in answering the following main research question:

RQ 2 How to infer scores of so-called missing documents in data fusion?

To answer this main research question, we propose a data fusion algorithm, TimeRA (time-aware rank aggregation), where we apply matrix factorization to model both result lists (called "users" in CF) and documents ("products" in CF) as a mixture of latent factors ("interests" in CF), such that the rank scores of missing documents ("the rating of unobserved products" in CF) in a result list for fusion can be inferred. In our data fusion algorithm, we define a list-document rank score matrix, factorize this matrix, and utilize the factorized list-specific and document-specific matrices to assign scores to missing documents.

Our results show that TimeRA is able to effectively fuse result lists, subject to real-time search requirements, running almost as fast as standard fusion methods such as CombSUM (Fox and Shaw, 1994), and outperform both time-sensitive microblog search methods and competing fusion algorithms.

Our contributions in this chapter can be summarized as follows:

Table 5.1: Additional notation used in TimeRA (cf. Table 2.1).

Notation	Gloss
$\mathbf{R}$	$m \times \lvert\mathcal{C}_\mathbf{L}\rvert$ list-post rank score matrix
A	number of latent factors
$\mathbf{S}$	matrix used for inferring latent topics for $\mathbf{L}$
$\mathbf{V}$	matrix used for inferring latent topics for $\mathcal{C}_\mathbf{L}$
$\mathbf{S}_i$	a column vector in S used for aspects of L_i
$\mathbf{V}_j$	a column vector in V used for aspects of d_j
$\mathfrak{R}_{ij}$	inferred rank score of post $d_j \in \mathcal{C}_\mathbf{L} \setminus L_i$
$d_k L_i d_j$	d_k ranks higher than d_j in L_i
I_{ij}	indicate whether $d_j \in L_i$
β	relative weight of burst information in TimeRA

i. We propose a data fusion-based algorithm for microblog search, TimeRA, that outperforms traditional and state-of-the-art unsupervised and supervised data fusion methods as well as the best single result list that it aggregates.

ii. We examine the relative contributions of the main ingredients of TimeRA (burst detection and score inference) and find that burst detection is effective in helping data fusion and that inferring scores for missing documents during fusion boosts performance as well.

iii. We provide a detailed analysis of the performance of TimeRA and offer a number of examples where we observe the effect hypothesized in Fig. 4.1 in Chapter 4, i.e., of posts in bursts having their rank score boosted.

iv. We show that the effect of TimeRA on ranking microblog posts is different from the effect of existing time-sensitive microblog search models.

v. And, finally, we study the efficiency of TimeRA and show that it meets real-time search requirements.

In the remainder of the chapter, we detail our approach in Section 5.1; Section 5.2 describes experimental setup; Section 5.3 presents the results; finally, Section 5.4 concludes the chapter.

5.1 Time-Aware Data Fusion

We detail our time-aware data fusion algorithm in this section; Section 5.1.1 details our fusion method, TimeRA, and Section 5.1.2 contains a brief complexity analysis of TimeRA. The way we detect bursts in our TimeRA algorithm is the same as that in BurstFuseX in Chapter 4. See 4.1.2 for details of how to detect bursts. To be able to present data fusion methods in a uniform way, we first introduce and summarize additional notation in Table 5.1 (other notation used throughout the thesis can be found in Table 2.1).

Algorithm 2: TimeRA: Time-Aware data fusion.

Input : A query q, a number of ranked lists of posts to be fused, $\mathbf{L}$=$\{ L_1, L_2, \ldots, L_m \}$, the combined set of posts $\mathcal{C}_\mathbf{L} := \bigcup_{i=1}^{m} L_i$, learning rate δ.
Output: A final data fusion list of posts L_f.

1 $L_f = \varnothing$, initialize $\mathbf{R}$ by (2.1), Initialize $\mathbf{S}$ and $\mathbf{V}$ with random values, let $m = |\mathbf{L}|$ and $t = 0$;
2 **for** $j = 1, 2, \cdots, n$ **do**
3 $\quad f_{\mathrm{TimeRA}}(d_j|q) = 0$;
4 Generate a set of bursts $\mathcal{B}$ (See Chapter 4);
5 **repeat**
6 $\quad$ **for** $i = 1, 2, \cdots, m$ **do**
7 $\quad\quad \mathbf{S}_i^{(t+1)} = \mathbf{S}_i^{(t)} - \delta \frac{\partial C_2}{\partial \mathbf{S}_i^{(t)}}$ based on (5.13);
8 $\quad$ **for** $j = 1, 2, \cdots, n$ **do**
9 $\quad\quad \mathbf{V}_j^{(t+1)} = \mathbf{V}_j^{(t)} - \delta \frac{\partial C_2}{\partial \mathbf{V}_j^{(t)}}$ based on (5.13);
10 $\quad t = t + 1$;
11 **until** C_2 *given by* (5.10) *converges;*
12 $\mathbf{S} = \mathbf{S}^{(t)}$, $\mathbf{V} = \mathbf{V}^{(t)}$;
13 **for** $j = 1, 2, \cdots, n$ **do**
14 $\quad$ Obtain $f_{\mathrm{TimeRA}}(d_j|q)$ score for d_j via $\mathbf{S}$ and $\mathbf{V}$ by (5.12);
15 Construct final fused list L_f via decreased scores of $f_{\mathrm{TimeRA}}(d|q)$.

5.1.1 The fusion method

Next we detail our time-aware data fusion method, TimeRA. TimeRA utilizes matrix factorization techniques. The input of the matrix factorization in TimeRA is an $m \times n$ matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ which we call a *list-document* matrix; here, again, m is the number of result lists and n is the number of posts to be fused, i.e., $n = |\mathcal{C}_\mathbf{L}|$. $\mathbf{R}$ is initialized by (2.1), viz., its elements R_{ij} are defined by (2.1). The output of the matrix factorization consists of two new matrices $\mathbf{S} \in \mathbb{R}^{A \times m}$ and $\mathbf{V} \in \mathbb{R}^{A \times n}$, obtained by factorizing $\mathbf{R}$, which we call the *factor-list* matrix and the *factor-post* matrix, respectively. Here, A is the number of latent factors. After obtaining $\mathbf{S}$ and $\mathbf{V}$, we can infer the normalized scores of missing posts, based on which we arrive at the fusion scores of all the posts.

We present TimeRA in Algorithm 2. We first provide a high-level walkthrough. To begin, it sets matrices $\mathbf{S} \in \mathbb{R}^{A \times m}$, $\mathbf{V} \in \mathbb{R}^{A \times n}$ with random values and $f_{\mathrm{TimeRA}}(d|q)$ with the value 0 (lines 1–3 in Algorithm 2). Let $\mathbf{S}_i$ be the i-th column of $\mathbf{S}$, meant to get the latent factors of the list L_i after matrix factorization, and let $\mathbf{V}_j$ be the j-th column of $\mathbf{V}$, meant to get latent factors of post d_j after matrix factorization. After detecting bursts (line 4), TimeRA utilizes burst information and tries to get the final $\mathbf{S}$ and $\mathbf{V}$ by performing a gradient descent process on a cost function C_2 (lines 5–12). To this end, fusion scores $f_{\mathrm{TimeRA}}(d|q)$ of $d \in \mathcal{C}_\mathbf{L}$ can be obtained by $\mathbf{S}$ and $\mathbf{V}$ (lines 13–14). The defined cost function plays an important role: (1) it tries to keep the original rank scores of posts that rank high in many of the lists to be fused, (2) it rewards posts that rank low in a few lists but in the vicinity of bursts, and (3) it gets the latent factors of both the lists

and the posts such that missing posts' rank scores can be inferred.

Our matrix factorization approach in TimeRA seeks to approximate the list-document matrix $\mathbf{R}$ by a multiplication of A latent factors,

$$\mathbf{R} \approx \mathbf{S}^\top \mathbf{V} \tag{5.1}$$

To obtain $\mathbf{S}$ and $\mathbf{V}$, traditionally, the Singular Value Decomposition (SVD) method (Kurucz et al., 2007) is utilized to approximate the list-document rank matrix $\mathbf{R}$ by minimizing:

$$\mathbf{S}, \mathbf{V} = \arg\min_{\mathbf{S},\mathbf{V}} \frac{1}{2} ||\mathbf{R} - \mathbf{S}^\top \mathbf{V}||_F^2, \tag{5.2}$$

where $||\cdot||_F^2$ denotes the Frobenius norm. Due to the definition that $R_{ij} = 0$ if d_j is a missing document, i.e., $d_j \in \mathcal{C}_\mathbf{L} \setminus L_i$, we only need to factorize the observed scores in $\mathbf{R}$ so that (5.2) changes to:

$$\mathbf{S}, \mathbf{V} = \arg\min_{\mathbf{S},\mathbf{V}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} (R_{ij} - \mathbf{S}_i^\top \mathbf{V}_j)^2, \tag{5.3}$$

where I_{ij} is an indicator function that is equal to 1 if d_j appears in L_i and equal to 0 otherwise, $\mathbf{S}_i \in \mathbb{R}^{A \times 1}$ is a column vector in $\mathbf{S}$ that serves to get latent factors of L_i, and, similarly, $\mathbf{V}_j \in \mathbb{R}^{A \times 1}$ is a column vector in $\mathbf{V}$ that serves to get latent factors of d_j. As $R_{ij} \in [0, 1]$ according to (2.1), (5.3) can be rewritten as:

$$\mathbf{S}, \mathbf{V} = \arg\min_{\mathbf{S},\mathbf{V}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} (R_{ij} - g(\mathbf{S}_i^\top \mathbf{V}_j))^2, \tag{5.4}$$

where $g(x)$ is the logistic function defined by $g(x) = 1/(1 + \exp(-x))$, which makes it possible to bound the range of $\mathbf{S}_i^\top \mathbf{V}_j$ within the same range $[0, 1]$ by R_{ij}.

In order to avoid overfitting, two regularization terms are added to (5.4):

$$\begin{aligned} \mathbf{S}, \mathbf{V} = \arg\min_{\mathbf{S},\mathbf{V}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} (R_{ij} - g(\mathbf{S}_i^\top \mathbf{V}_j))^2 \\ + \frac{\lambda_1}{2} ||\mathbf{S}||_F^2 + \frac{\lambda_2}{2} ||\mathbf{V}||_F^2, \end{aligned} \tag{5.5}$$

where $\lambda_1, \lambda_2 > 0$. For a probabilistic interpretation with Gaussian observation noise for (5.5) we ref to (Salakhutdinov and Mnih, 2008b). To reduce the model's complexity we follow (Kurucz et al., 2007; Ma et al., 2011a; Salakhutdinov and Mnih, 2008a,b) and set $\lambda_1 = \lambda_2$ in all experiments we conduct in the chapter.

The cost function defined by (5.5) punishes posts equally when they shift from the original rank scores R_{ij}. However, a post ranked higher should be punished more than posts ranked lower if they shift from the original rank scores: higher ranked posts are more likely to be relevant so that keeping their original rank scores will be of more value. Thus, we modify (5.5) to obtain the following cost function C_1:

$$\begin{aligned} \mathbf{S}, \mathbf{V} = \arg\min_{\mathbf{S},\mathbf{V}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} \cdot w(d_j | L_i) (R_{ij} - g(\mathbf{S}_i^\top \mathbf{V}_j))^2 \\ + \frac{\lambda_1}{2} ||\mathbf{S}||_F^2 + \frac{\lambda_2}{2} ||\mathbf{V}||_F^2, \end{aligned} \tag{5.6}$$

where $w(d_j|L_i)$ is a rank punishment weighting function defined for d_j given L_i. Specifically, we define $w(d_j|L_i)$ as:

$$w(d_j|L_i) = \begin{cases} \frac{1}{2^{\text{rank}(d_j|L_i)-1}} & d_j \in L_i \\ 0 & d_j \notin L_i. \end{cases} \tag{5.7}$$

Our next step is to bring in burst information. After detecting a set of bursts, we introduce the following item into the cost function C_1 to boost the relevance of posts by burst information, such that:

$$\begin{aligned} \mathbf{S}, \mathbf{V} = \arg\min_{\mathbf{S},\mathbf{V}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{b \in \mathcal{B}} \frac{1}{|\{d_k \in b : d_k L_i d_j\}|} \\ \cdot \sum_{\substack{d_k \in b \\ d_k L_i d_j}} I_{ij} I_{ik} r(d_j|d_k) w(d_k|L_i)(R_{ik} - g(\mathbf{S}_i^\top \mathbf{V}_j))^2, \end{aligned} \tag{5.8}$$

where $\mathcal{B}$ is a set of detected bursts, b is a burst in $\mathcal{B}$, $d_k L_i d_j$ means d_k is ranked higher than d_j in list L_i, $|\{d_k \in b : d_k L_i d_j\}|$ is the total number of posts in the burst b that are ranked higher than the post d_j in L_i, and $r(d_j|d_k)$ is the "reward" score for d_j from d_k if d_j is within or near the burst b to which d_k belongs. In (5.8), $d_k L_i d_j$ indicates that only the posts ranked higher than the post d_j to be rewarded are capable of boosting the relevance of d_j.

If d_j, the post to be rewarded, is generated at (almost) the same time as the highly relevant post d_k, it will be rewarded more than posts that are further away in time from d_k, which is measured by $r(d_j|d_k)$ in (5.8). Accordingly, we define $r(d_j|d_k)$ based on a normal distribution as:

$$r(d_j|d_k) = \exp\left\{-\frac{(t_{d_j} - t_{d_k})^2}{2\sigma_b^2}\right\}, \tag{5.9}$$

where t_{d_j} and t_{d_k} are the timestamps of post d_j and d_k, respectively, and σ_b is the standard deviation of timestamps in burst b:

$$\sigma_b^2 = \frac{\sum_{i=1}^{n_b} \{i - \frac{n_b+1}{2}\}^2}{n_b} = \frac{n_b^2 - 1}{12},$$

where n_b is the number of different timestamps of posts in b, and the detailed derivation of which can be referred to (4.8) in Chapter 4. According to (5.9), the more likely d_j is generated at the same time with d_k, the larger the score $r(d_j|d_k)$ is, resulting in a bigger reward for d_j from d_k.

We are almost ready now to define the cost function C_2 that we use in TimeRA. We integrate the original rank score of the posts in the result lists ((5.6)) with rewards for posts generated in the vicinity of bursts ((5.8)). To this end, we substitute (5.8) into (5.6).

Therefore, our cost function C_2 for TimeRA is defined as:

$$\begin{aligned}\mathbf{S},\mathbf{V} = \arg\min_{\mathbf{S},\mathbf{V}} & \frac{1-\beta}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij} w(d_j|L_i)(R_{ij} - g(\mathbf{S}_i^\top \mathbf{V}_j))^2 \\ & + \frac{\beta}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{b\in\mathcal{B}} \frac{1}{|\{d_k \in b : d_k L_i d_j\}|} \sum_{\substack{d_k\in b \\ d_k L_i d_j}} I_{ij} I_{ik} \\ & \cdot r(d_j|d_k)\cdot w(d_k|L_i)(R_{ik} - g(\mathbf{S}_i^\top \mathbf{V}_j))^2 \\ & + \frac{\lambda_1}{2}||\mathbf{S}||_F^2 + \frac{\lambda_2}{2}||\mathbf{V}||_F^2,\end{aligned} \tag{5.10}$$

where $\beta \in [0,1]$ is a free parameter that governs the linear mixture.

In words, in response to a query q, TimeRA uses a two-component mixture model to score $d \in \mathcal{C}_\mathbf{L}$. The first component (the first term of (5.10)) tries to maintain the original rank scores R_{ij}. The second component (the second term of (5.10)), which uses bursts for posts, "rewards" d if it is strongly associated with the posts in the bursts. Clearly, if $\beta = 0$ in (5.10), TimeRA will only try to maintain the original rank scores, i.e., (5.6).

A local minimum of the objective function given by (5.10) can be found by performing gradient descent in both $\mathbf{S}_i$ and $\mathbf{V}_j$. The same can apply to (5.2), (5.3), (5.4), (5.5), and (5.6). The algorithm for obtaining $\mathbf{S}$ and $\mathbf{V}$ is straightforward: we first randomly initialize $\mathbf{S}$ and $\mathbf{V}$, then iteratively update these two matrices based on their gradients until the value of the cost (objective) function converges. The derivation of these equations are included in Appendix 5.A.

After optimizing (5.10), posts $d_j \in L_i$ they will end up with a score $g(\mathbf{S}_i^\top \mathbf{V}_j) = R_{ij}$ + "some reward." Unlike previous work that assigns 0 to missing documents $d_j \in \mathcal{C}_\mathbf{L} \setminus L_i$ (Efron, 2011; Fox and Shaw, 1994; Kozorovitsky and Kurland, 2011; Sheldon et al., 2011; Tsai et al., 2008; Wu, 2012), we infer a rank score $\mathfrak{R}_{L_i d_j}$ for missing posts d_j as:

$$\mathfrak{R}_{ij} = \begin{cases} g(\mathbf{S}_i^\top \mathbf{V}_j) & \text{if } g(\mathbf{S}_i^\top \mathbf{V}_j) \le \min(R_{id}) \\ \min(R_{id}) & \text{if } g(\mathbf{S}_i^\top \mathbf{V}_j) > \min(R_{id}) \end{cases}, \tag{5.11}$$

where $\min(R_{id})$ is the minimum rank score of the lowest ranked post that appears in L_i as computed by (2.1) in Chapter 2. As shown in (5.11), if the inferred rank score $g(\mathbf{S}_i^\top \mathbf{V}_j)$ is smaller than the minimum rank score, we maintain the inferred score for that post. However, if the inferred rank score is greater than the minimum rank score, we give up the inferred score and let $\mathfrak{R}_{ij} = \min(R_{id})$, as $d_j \notin L_i$ means that it should at least be ranked lower than any post $d \in L_i$.

The final data fusion score for $d_j \in \mathcal{C}_\mathbf{L}$ is obtained by:

$$f_{\text{TimeRA}}(d_j|q) = \sum_{\substack{i=1 \\ d_j \in L_i}}^{m} g(\mathbf{S}_i^\top \mathbf{V}_j) + \sum_{\substack{i=1 \\ d_j \in \mathcal{C}_\mathbf{L}\setminus L_i}}^{m} \mathfrak{R}_{ij}. \tag{5.12}$$

The final data fusion score for a post consists of two parts, i.e., scores for lists L_i it appears in (the first term in (5.12)) and scores for lists it does not appear in (the second

term in (5.12)), as inferred by (5.11). Finally, the final fused list L_f produced in response to q is obtained by ranking posts in decreasing order of $f_{\mathrm{TimeRA}}(d_j|q)$. We call the model defined by (5.12), *TimeRA*, and the variant that ignores inferred rank information $\mathfrak{R}_{ij}$ in (5.12) is called *TimeRA-Infer* ("TimeRA minus Infer").

5.1.2 Analysis of time-aware data fusion

The main computation of TimeRA is detecting bursts and its gradients against matrices $\mathbf{S}$ and $\mathbf{V}$. Our burst detection has computational complexity $O(n)$ because the detection algorithm runs in linear time (Ruzzo and Tompa, 1999), and the gradients of C_2 have computational complexity of $O(e \times m \times n \times A)$. In practice, the number of result lists to be fused, m, the number of posts to be aggregated, n, the number of latent factors, A, and the number of epochs, e, are all quite small. Therefore, the data fusion procedure can run in real-time, as we will see in the experiments presented in Section 5.3.6.

In the limiting case, TimeRA reverts back to CombSUM. That is, if we set $\beta = 0$ (ignoring burst information), set the weight function $w(d|L_i) = 1$ for all documents d and lists L_i, and, moreover, set $\lambda_1 = \lambda_2 = 0$, and do not try to infer the rank score of posts, then $g(\mathbf{S}_i^\top \mathbf{V}_j) = R_{ij}$ after performing gradient descent in (5.10). Our experimental results below show that the performance of TimeRA is almost the same as CombSUM when $\beta = 0$. Note that $\lambda_1, \lambda_2 \neq 0$ in the experiments.

5.2 Experimental Setup

In this section, we list our research questions, describe the data set, specify our baselines, detail the metrics as well as our training and optimization setup, and describe the experiments.

5.2.1 Detailed research questions

We divide our main research question (RQ 2) into the following detailed research questions, and let these questions guide the remainder of the chapter:

RQ 2.1 Does TimeRA outperform traditional and state-of-the-art unsupervised or supervised data fusion methods, the best single result list, and the BurstFuseX method introduced in Chapter 4? (Section 5.3.1 for answer.)

RQ 2.2 What are the relative contributions of the main ingredients of TimeRA (burst detection and score inference)? (Section 5.3.2 for answer.)

RQ 2.3 What is the effect of using burst information in TimeRA (i.e., what is the impact of the parameter β in (5.10))? (Section 5.3.3 for answer.)

RQ 2.4 What is the effect of the number of lists to be aggregated in TimeRA? (Section 5.3.4 for answer.)

RQ 2.5 Can we observe the hypothesized effect sketched in Fig. 4.1, i.e., posts in bursts being rewarded? (Section 5.3.5 for answer.)

RQ 2.6 Does TimeRA meet real-time search requirements? (Section 5.3.6 for answer.)

RQ 2.7 Does TimeRA beat BurstFuseX and other time-sensitive microblog search models? (Section 5.3.7 for answer.)

5.2.2 Baselines and evaluation

We compare TimeRA to 6 fusion baselines: 2 traditional unsupervised methods, i.e., CombSUM, CombMNZ, 2 start-of-the-art cluster-based fusion methods, ClustFuseCombSUM and ClustFuseCombMNZ (Kozorovitsky and Kurland, 2011), a start-of-the-art supervised method, λ-Merge (Sheldon et al., 2011), in which we integrate temporal features, and our proposed BurstFuseX presented in Chapter 4. As TimeRA utilizes temporal information, besides BurstFuseX we also compare TimeRA to 4 additional state-of-the-art time-sensitive microblog search algorithms (also see Section 4.2.2 in Chapter 4): TBLM (Li and Croft, 2003), LM-T(qe) (Massoudi et al., 2011), DIRECT-BM25 (mean) (Dakka et al., 2012) and TSF+QDRM (Miyanishi et al., 2013b). To build the index of the dataset that some of our baselines require, we apply Porter stemming, tokenization, and stopword removal (using INQUERY lists) to posts using the Lemur toolkit.[1] Features used in λ-Merge, including those described in Appendix 4.A in Chapter 4 as well as additional time-sensitive features, i.e., retrieval scores from 4 time-sensitive retrieval baselines: TBLM, LM-T(qe), DIRECT-BM25 (mean), and TSF+QDRM.

For performance evaluation we use the official TREC Microblog 2011 metric, p@30. We also report on p@5, p@10, p@15 and MAP scores. MAP scores are of special interest to us: we hypothesize that TimeRA has both a precision and recall-enhancing effect and we use MAP to measure this.

A single free parameter in (5.10), β ($\in \{0, 0.1, \ldots, 1\}$), is incorporated in TimeRA, which is set using leave-one-out cross validation performed over the entire set of 49 queries. The performance of MAP is optimized in the learning phase. In other words, the performance for a query is attained using a value of β that maximizes MAP performance over all other queries. The same optimization strategy is used for one of our baselines, cluster-based fusion. Other baselines do not incorporate free parameters. Following (Ma et al., 2011b), we set the parameters $\lambda_1 = \lambda_2 = 0.001$.

5.2.3 Experiments

We report on 7 main experiments in this chapter aimed at understanding (1) the performance of TimeRA in general via sampling lists and fusing them; (2) the contribution of the main ingredients in TimeRA; (3) the performance of TimeRA with increasing numbers of runs to be fused; (4) query level performance; (5) TimeRA's efficiency; (6) the effect of inferring rank scores of posts by TimeRA; (7) the performance of TimeRA against temporal retrieval models. The experiments are carried out on the microblog search collection. Details of the microblog search collection can be found in Section 3.2.2

As in Chapter 4, to understand the overall performance of TimeRA, we sample $\sim$10% from the ranked lists produced by participants in the TREC 2011 Microblog track based on the lists' p@30 distribution: 18 out of the runs submitted to the TREC 2011 Microblog track, 6 with p@30 scores between 0.20 and 0.30 (Class 3), 6 between 0.30 and 0.40 (Class 2), and 6 over 0.40 (Class 1). We also randomly choose two runs from each class

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

to construct Class 4; see Table 4.2. The runs in Class 1 are the 6 best runs in the TREC 2011 Microblog track. In every class, we use run1, run2, run3, run4, run5 and run6 to refer to the runs in descending order of p@30.

Next, as in Chapter 4, to understand the contributions of the two main ingredients of TimeRA, viz., burst detection and inferring scores, we make comparisons among TimeRA, TimeRA-Infer and CombSUM. We also gradually increase the parameter β in (5.10) from 0.0 to 1.0 to see if burst information is helpful to boost fusion performance.

As in Chapter 4, to understand the effect of the number of lists being merged, we randomly choose $k = 2, 4, 6, \ldots, 36$ lists from the 184 lists and aggregate them. We repeat the experiments 20 times and report the average results and standard deviation. In order to understand the query-level performance of TimeRA, we provide a detailed comparison of its performance against the baseline methods. To determine whether TimeRA can respond to a given query in (near) real time, we again randomly fuse $k = 2, 6, 12, 18, 30$ lists for all 49 test queries and report the average time required. Finally, we compare TimeRA against state-of-the-art time-sensitive retrieval models that utilize time/burst information.

5.3 Results and Analysis

Section 5.3.1 and Section 5.3.2 show the results of fusing the sample lists, the contributions of burst detection and score inference in TimeRA, respectively; Section 5.3.3 analyzes the effect of using burst information; Section 5.3.4 shows the effect of the number of lists on the overall performance; Section 5.3.5 provides a topic-level analysis; Section 5.3.6 examines the run times. Finally, Section 5.3.7 compares TimeRA against time-sensitive models.

5.3.1 Fusing the sample lists

The performance of TimeRA and the 6 baselines is presented in Table 5.2, with numbers based on the $\sim$10% sample mentioned in Section 5.2.3. The performance of all the fusion methods is better than that of the best performing result list that is used in the merging process (run1) for all classes and on almost all metrics. Many of these improvements are statistically significant. More importantly, when fusing the top 6 result lists (Class 1), all of the p@30 scores generated by any data fusion method are higher than that of the best run in TREC 2011 Microblog track (0.4551), especially for TimeRA, which achieves 0.5531. These findings attest to the merits of using data fusion methods for microblog search. The performance of TimeRA proposed in this chapter has the best retrieval performance in most cases in terms of all the metrics under consideration.

It is also worth noting in Table 5.2 that, as in Chapter 4, in almost all cases, the cluster-based method does not beat the standard fusion method that it integrates, and the performance differences between the two are usually not significant. As we suggested in Chapter 4, the reason behind this may be that it is challenging to do clustering in a microblog environment, with limited amounts of text and very creative language usage. In most cases, the BurstFuseX methods can beat the standard fusion method it integrates in terms of MAP, p@5, p@10, p@15 and p@30.

The performance of TimeRA is better than that of the baseline methods, and almost all of the differences are substantial and statistically significant. The performance of λ-Merge is almost the same as that of CombSUM, CombMNZ and the cluster-based methods when fusing the lists in Class 2, but in the other classes the performance tends to be a bit below that of the other methods, on all metrics. This may be due to overfitting.

Interestingly and confirming an observation from Chapter 4, the higher the quality of the result lists that are being aggregated, the bigger the improvements that can be observed in Table 5.2. For instance, the p@30 scores after fusion are highest in Class 1 followed by those in Class 2 and Class3, and the quality of Class 1 is best followed by Class 2 and Class 3, respectively. The p@30 fusion scores in Class 4 are almost the same as those in Class 2, as some of the lists' scores in Class 4 are better than those in Class 2.

5.3.2 Contributions of the main ingredients

Next, we compare the relative contributions of the main ingredients of TimeRA against the very well-performed baseline, viz., CombSUM: burst detection and score inference. The effect of burst detection in TimeRA can be seen through comparisons between TimeRA-Infer and CombSUM; the effect of score inference can be seen through comparisons between TimeRA and TimeRA-Infer in Fig. 5.1.

Interestingly, in Fig. 5.1 there are large gaps in performance between TimeRA-Infer and CombSUM in terms of all of metrics; all improvements are statistically significant. This illustrates that burst detection makes an important contribution to the performance of data fusion. When we compare TimeRA and TimeRA-Infer in Fig. 5.1, we see that the performance of TimeRA-Infer in terms of p@5 is almost the same as that of TimeRA, while in terms of p@10 and p@30, TimeRA has some small advantages over TimeRA-Infer—some of these improvements are statistically significant. This observation confirms that inferring scores for posts during fusion can boost performance as well. It also shows, however, that enhancing the performance of p@k becomes easier for larger values of k. This is because the cost of boosting the performance (i.e., changing the original rank score to be higher) is smaller when the posts are ranked lower. TimeRA is unable to beat TimeRA-Infer in terms of p@5 (.6939 for both), but TimeRA does boost the p@30 performance (.5531$^{\triangle}$ for TimeRA vs .5405 for TimeRA-Infer).

As burst information is such an important contributor to the performance of TimeRA, we analyze it further in Section 5.3.3.

5.3.3 The use of burst information

Next we examine the effect of using different amounts of burst information or cluster information in our time-aware fusion or cluster-based methods, respectively. What is the impact of the free parameter β in (5.10) and in the cluster-based methods? Fig. 5.2 depicts the MAP performance curves for all data fusion methods when fusing the lists in Class 1, Class 2, Class 3 and Class 4, respectively. For $\beta = 0$, TimeRA almost amounts to CombSUM, while the cluster-based methods are the same as the standard fusion methods they incorporate, e.g., ClustFuseCombSUM has no difference with CombSUM in this case; more weight is put on burst information and cluster information with higher values of β in TimeRA and the cluster-based methods, respectively. For $0 < \beta < 1$, both

Table 5.2: Retrieval performance on the ∼10% sample lists. Boldface marks the best result per metric; a statistically significant difference between TimeRA and the best baseline method is marked in the upper right hand corner of the TimeRA score. A significant difference with run1 for each method is marked in the upper left hand corner using the same symbols. None of the differences between the cluster-based method and the standard method it incorporates are statistically significant.

	Class 1					Class 2				
	MAP	p@5	p@10	p@15	p@30	MAP	p@5	p@10	p@15	p@30
run1	.2210	.5918	.5673	.5347	.4551	.1457	.4612	.4143	.3714	.3571
run2	.2690	.5959	.5796	.5442	.4537	.1886	.4776	.4347	.3878	.3463
run3	.2318	.5755	.5367	.5034	.4401	.1525	.4041	.4143	.3878	.3408
run4	.2058	.5714	.5367	.4939	.4211	.1376	.3959	.3939	.3796	.3218
run5	.2575	.5673	.4980	.4721	.4211	.1688	.3878	.3633	.3605	.3136
run6	.2098	.5469	.5102	.4694	.4095	.1820	.4122	.3796	.3619	.3027
CombSUM	▲.3404	▲.6245	.5816	.5524	▲.4966	▲.2625	▲.5306	▲.4531	▲.4286	▲.3735
ClustFuseCombSUM	▲.3398	▲.6240	.5802△	.5503	▲.4899	▲.2612	▲.5287	△.4500	▲.4213	▲.3686
BurstFuseCombSUM	▲.3563	▲.6163	▲.5959	▲.5878	▲**.5578**	▲.2651	△.4898	▲.4694	▲.4553	▲.4312
CombMNZ	▲.3385	▲.6245	.5755	.5524	▲.5020	▲.2581	▲.5347	△.4592	▲.4354	▲.3789
ClustFuseCombMNZ	▲.3355	▲.6231	.5748	.5502△	▲.4987	▲.2560	▲.5330	▲.4523	▲.4311	▲.3731
BurstFuseCombMNZ	▲.3528	▲.6286	▲.5959	▲.5918	▲.5517	▲.2587	▲.5061	▲.4735	▲.4567	▲.4242
λ-Merge	▲.3245	▲.6213	.5734	.5456	△.4833	▲.2573	▲.5634	▲.4952	▲.4463	▲.3901
BurstFuseλ-Merge	▲.3421	▲.6243	.5942	▲.5837	▲.5238	▲.2723	▼.5483	▲.5138	▲.4672	▲.4132
TimeRA	▲**.3834**▲	▲**.6939**▲	▲**.6510**▲	▲**.6259**▲	▲.5531	▲**.3037**▲	▲**.6327**▲	▲**.5551**▲	▲**.5211**▲	▲**.4320**

	Class 3					Class 4				
	MAP	p@5	p@10	p@15	p@30	MAP	p@5	p@10	p@15	p@30
run1	.1661	.4041	.3408	.2898	.2122	.2058	.5714	.5367	.4939	.4211
run2	.0997	.3429	.3000	.2653	.2095	.2098	.5469	.5102	.4694	.4095
run3	.1636	.3959	.3122	.2571	.2041	.1376	.3959	.3939	.3796	.3218
run4	.0753	.3265	.2735	.2585	.2034	.1820	.4122	.3796	.3619	.3027
run5	.0571	.2980	.2551	.2408	.2020	.1636	.3959	.3122	.2571	.2041
run6	.0994	.3510	.2735	.2408	.2016	.0753	.3265	.2735	.2585	.2034
CombSUM	▲.2150	▲.4857	▲.4327	▲.3837	▲.2952	▲.2795	▲.6122	▲.5327	▲.4721	▽.3918
ClustFuseCombSUM	▲.2142	▲.4804	▲.4267	▲.3806	▲.2882	▲.2741	▲.6088	△.5297	.4674	▼.3865
BurstFuseCombSUM	△.2283	▲.4408	▲.4184	▲.3973	▲**.3318**	▲.2863	▲.5633	▲.5449	▲.5011	▲.4312
CombMNZ	▲.2187	▲.4898	▲.4327	▲.3932	▲.2973	▲.2794	▲.6000	▲.5449	▲.4830	.4048
ClustFuseCombMNZ	▲.2151	▲.4872	▲.4265	▲.3886	▲.2894	▲.2744	▲.5975	▲.5407	△.4782	.3958
BurstFuseCombMNZ	▲.2313	▲.4531	▲.4327	▲.4122	▲.3236	▲.2883	▲.5796	▲.5469	△.5043	.4387
λ-Merge	▲.2125	▲.5057	▲.4373	▲.3827	▲.2933	▲.2753	▲.6046	▲.5387	▲.4918	.4047
BurstFuseλ-Merge	▲.2382	▲.4937	▲.4389	▲.4012	▲.3115	▲.2814	▼.5883	▲.5412	▲.5174	.4221
TimeRA	▲**.2491**▲	▲**.5347**▲	▲**.4694**▲	▲**.4122**▲	▲.3293	▲**.3210**▲	▲**.6735**▲	▲**.5918**▲	▲**.5429**▲	▲**.4347**

the CombSUM scores of posts and burst information are utilized for aggregating lists in TimeRA.

In each of our four classes of runs, when aggregating lists, the MAP scores of TimeRA where burst information is used ($\beta > 0$) are always higher than that of any other fusion method. In Class 1 and Class 4 the gain increases gradually as the weight of burst information increases. These findings attest to the merits of using burst information to boost the performance in fusing ranked lists for microblog search. Putting more weight on cluster information in the cluster-based methods hurts performance in many cases.

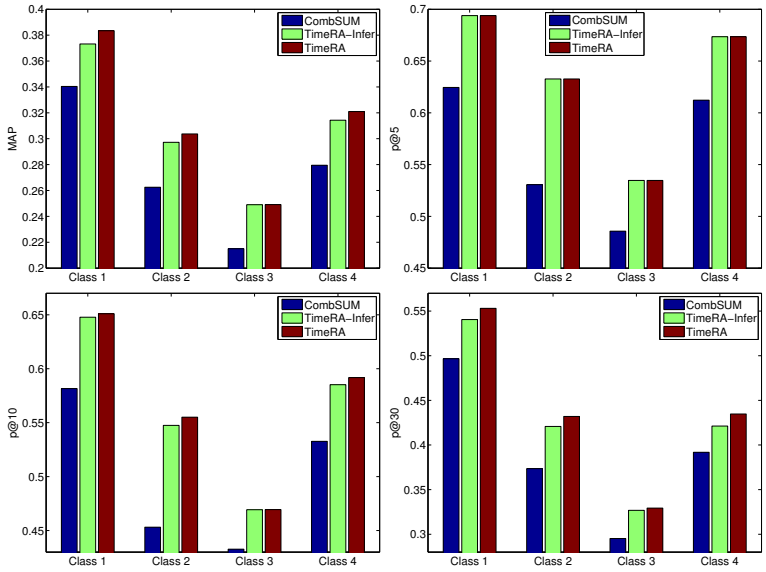


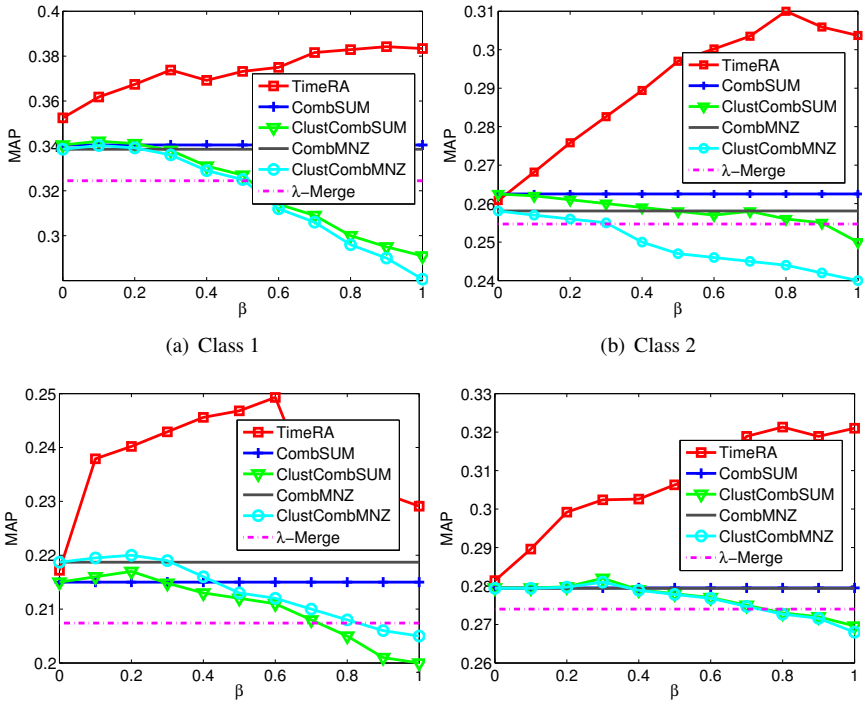
Figure 5.1: Retrieval performance of CombSUM, TimeRA-Infer (without using score inference for posts) and TimeRA in terms of MAP, p@5, p@10, and p@30 when fusing the runs in the 4 classes. Note that figures should be viewed in color.

5.3.4 Effect of the number of lists being merged

We explore the effect of varying the number of lists to be merged on the performance of TimeRA. Fig. 5.3 shows the fusion results of randomly sampling $k \in \{2, 4, 6, \ldots, 36\}$ lists from the 184 lists. For each k, we repeat the experiment 20 times and report on the average scores. We use CombSUM as a representative example for comparisons with TimeRA; the results of other baseline methods are worse or qualitatively similar to those of CombSUM.

From Fig. 5.3 we can see that TimeRA performs better than CombSUM over all performance evaluation metrics no matter how many lists are fused. For both precision metrics (p@5 and p@30) we find that as long as the number of lists ≤ 10, the performance of both TimeRA and CombSUM gradually increases as the number of lists to be merged increases. The increases level off when the number of lists exceeds 12. For MAP we find that performance keeps increasing until we fuse 26 lists; then, the performance increase levels off.

Interestingly, in Fig. 5.3 the improvement of TimeRA over CombSUM on p@5 becomes smaller when more lists are merged. When, for example, two lists are fused, the increase in p@5 of TimeRA over CombSUM is .1063 (.5861 for TimeRA vs .4798 for CombSUM). The performance increase, however, drops to only .0281 (.6712 for TimeRA vs .6431 for CombSUM) for 36 lists. Looking at the other metrics, which take a larger part of the merged result into account (p@30 and especially MAP), the gap remains.



(a) Class 1 (b) Class 2

(c) Class 3 (d) Class 4

Figure 5.2: Effect of varying the value of β on the MAP performance of six fusion methods, for lists in (a) Class 1, (b) Class 2, (c) Class 3 and (d) Class 4. More weight is put on burst information and on cluster information with higher values of β in TimeRA and the cluster-based methods, respectively. Note: the figures are not to the same scale.

5.3.5 Query-level analysis

As in Chapter 4, here we also take a closer look at per test query improvements of TimeRA over other runs. For brevity, we only consider CombSUM as a representative and we only consider runs in Class 1. The results of TimeRA against CombSUM for other classes of runs and for other baseline methods including the proposed BurstFuseX methods in Chapter 4 are qualitatively similar. Fig. 5.4 shows per query performance differences in terms of AP, p@5, p@10 and p@30, respectively, between TimeRA and CombSUM. TimeRA displays both a precision and recall enhancing effect (with increases in precision oriented metrics as well as in MAP). As the metric at hand considers a larger chunk of the result list, there are more instances where TimeRA outperforms CombSUM. This is due mainly to topics that are discussed only in very specific time intervals. Examples include queries MB010 (Egyptian protesters attack museum), MB011 (Kubica crash) and MB015 (William and Kate fax save-the-date) etc. For such queries we found evidence of the intuition depicted in Fig. 4.1: posts that are ranked low in a small number lists but that TimeRA pushes up because they are central to a burst. E.g., in response to query MB010, post #30354903104749568 is ranked near the bottom in

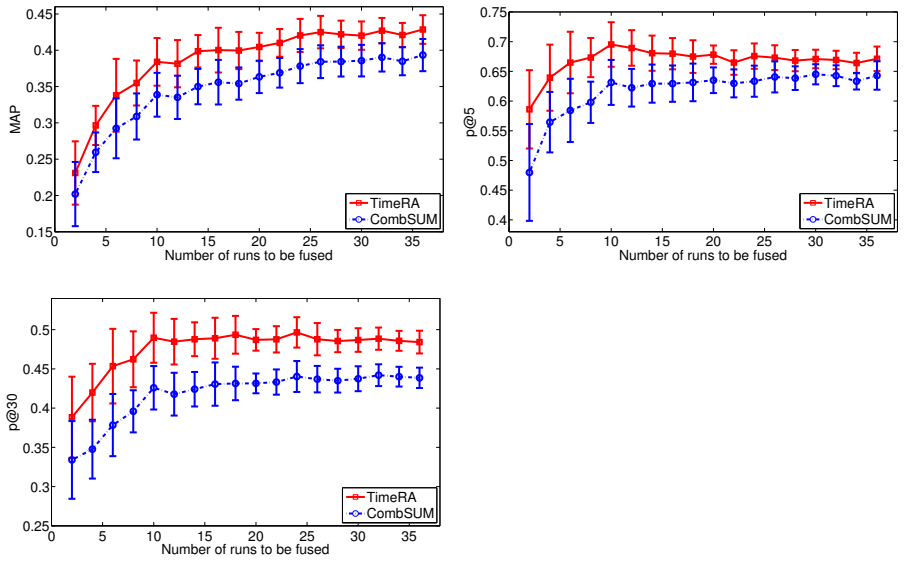


Figure 5.3: Effect on performance (in terms of MAP, p@5, and p@30) of the number of lists being merged. We plot averages and standard deviations. Note: the figures are not to the same scale.

just two lists (at ranks 26 and 27 in the runs clarity1 and DFReekLIM30, respectively). Many posts for the query were generated around the same time interval (Jan. 26–29) and are ranked high in many lists; post #30354903104749568 was also published around this time and ranked 6th in the merged list because of this.

Queries for which TimeRA cannot beat ComSUM tend to be quite general and unrelated to any specific time windows. Examples include queries MB023 (Amtrak train service), MB027 (reduce energy consumption) and MB029 (global warming and weather) etc. For a very small number of queries, TimeRA's performance is slightly worse than that of CombSUM. One reason that we observed for this phenomenon is that not all posts that are ranked low in a small number of lists but central to a burst need to be rewarded. An example here is query MB024 (Super Bowl, seats).

5.3.6 Run time comparisons

We now explore how fast TimeRA can merge result lists in response to a query. TimeRA is developed in C++ and the experiments are run on a 10.6.8 MacBook Pro computer with 4GB memory and a 2.3 GHz Intel core i5 processor. In Table 5.3, we randomly choose $k \in \{2, 6, 12, 18, 30\}$ lists from the 184 lists. For each k, we repeat the experiment 20 times and report the average run time per query (in seconds) that the fusion methods require. ClustSUM and ClustMNZ are short for ClustFuseCombSUM and ClustFuseCombMNZ, respectively in Table 5.3.

TimeRA does not run as fast as CombSUM or CombMNZ, but it manages to merge lists near real-time. TimeRA merges the lists within 0.1s when given 30 result lists and

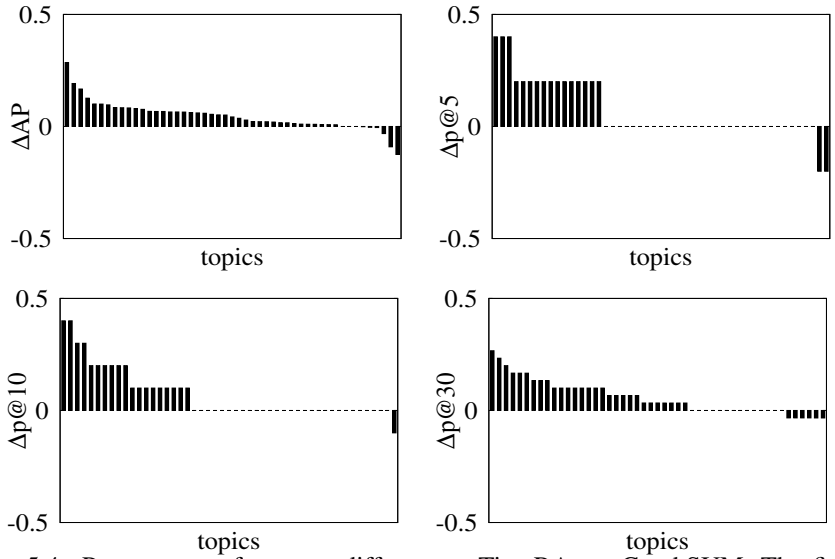


Figure 5.4: Per query performance differences, TimeRA vs. CombSUM. The figures shown are for the runs in Class 1, for MAP, p@5, p@10 and p@30 (left to right). A bar extending above the center indicates that TimeRA outperforms CombSUM, and vice versa.

within 0.01s when fusing two lists. As the number of lists to be fused increases, the time spent on fusing is linear for CombSUM, CombMNZ, TimeRA, and λ-Merge; for ClustSUM and ClustMNZ, the time increases dramatically with larger numbers of lists. When fusing 30 lists, ClustMNZ needs to spend 235.91s, although it only spends 1.03s on fusing two lists. The times clocked for CombSUM and CombMNZ are similar, and likewise for those of ClustSUM and ClustMNZ.

5.3.7 Effect of fusing time-sensitive result lists

TimeRA uses temporal information in an essential way. How does it compare against retrieval models that explore temporal information? To answer this question, we again show 4 additional experimental results from 4 time-sensitive result lists (The same results of these 4 lists can be seen in Section 4.3.7), and explore the performance of the baseline fusion methods and TimeRA with each of which respectively fuses these 4 lists. These lists are generated by TBLM (Li and Croft, 2003), LM-T(qe) (Massoudi et al., 2011), DIRECT-BM25 (mean) (Dakka et al., 2012) and TSF+QDRM (Miyanishi et al., 2013b). We also make comparison between TimeRA and BurstFuseX in Table 5.4.

Table 5.4 shows the fusion result. Obviously, except ClustFuseCombMNZ and λ-Merge, fusion baselines and TimeRA outperform the best time-sensitive component run (TSF+QDRM) for all metrics and most of the improvements are statistically significant. This illustrates that exploring time information in data fusion has a different effect than utilizing time information in an individual ranking function, an effect that can lead to performance increases. One reason behind this is that posts within intervals in which many

Table 5.3: Time spent on fusing lists by different fusion methods. Recorded in seconds with standard deviations (std).

	Number of lists				
	2	6	12	18	30
CombSUM	3.06e–4	5.76e–4	1.03e–3	1.98e–3	3.37e–3
std	1.13e–5	2.57e–5	6.93e–5	6.49e–5	7.50e–5
CombMNZ	3.06e–4	5.76e–4	1.03e–3	1.99e–3	3.38e–3
std	1.13e–5	2.57e–5	6.93e–5	6.52e–5	7.01e–5
TimeRA	4.77e–3	1.69e–2	2.05e–2	4.56e–2	9.60e–2
std	7.60e–5	7.41e–5	2.20e–4	3.53e–4	3.60e–4
λ-Merge	1.15	3.82	7.78	12.03	20.74
std	1.22e–1	5.82e–1	8.03e–1	1.09	1.18
ClustSUM	1.03	4.12	37.56	88.21	235.91
std	9.87e–2	4.24e–1	1.26	3.54	13.08
ClustMNZ	1.03	4.12	37.56	88.21	235.91
std	9.87e–2	4.24e–1	1.26	3.54	13.08

relevant posts appear can only be confirmed to be relevant by gathering data from multiple lists, time-sensitive or not. The TimeRA method outperforms BurstFuseX and some improvements are statistically significant, which again illustrates the inferring scores of missing documents can help to improve retrieval performance.

5.4 Conclusion

The special nature of microblog posts, e.g., their limited length and their creative language usage, raises challenges for searching them. However, this special nature also provides unique algorithmic opportunities. In this chapter, we focus on utilizing time information to boost the performance of searching microblog posts. Specifically, we proposed a novel data fusion approach, TimeRA, that utilizes bursts and only rank information to aggregate result lists. TimeRA first detects bursts of posts across the lists utilizing original rank information of the posts, and then rewards posts that are ranked low in few lists but in the vicinity of a burst that contains higher ranked posts. It also infers the rank scores of missing posts by modeling lists and posts as a mixture of latent factors. Our experiments provide answers to the main research question raised at the beginning of this chapter:

RQ 2 How to infer scores of so-called missing documents in data fusion?

To answer the question, we continued to work with the TREC microblog tracks dataset and fuse the result lists submitted to the TREC using our proposed methods and the baseline methods. Our experimental results show that both utilizing burst information and score inference for data fusion can significantly enhance retrieval performance even

Table 5.4: Performance on 4 time-sensitive result lists. Boldface marks the better result per metric; the best score of component lists per metric is underline; a statistically significant difference between TimeRA and the best baseline is marked in the upper right hand corner of TimeRA score; a statistically significant difference with TSF+QDRM for each fusion method is marked in the upper left hand corner of the score; a statistically significant difference with BurstFuseCombSUM and TimeRA is marked in the lower right hand corner of the TimeRA score; the statistically significant difference with BurstFuseCombMNZ and TimeRA is the same as that between BurstFuseSUM and TimeRA.

	MAP	p@5	p@10	p@15	p@30
TSF+QDRM	.2834	.6220	.6856	.6279	.5368
DIRECT-BM25 (mean)	.2798	.6187	.6725	.6320	.5133
LM-T (qe)	.2346	.5836	.5648	.5178	.4471
TBLM	.2231	.5742	.5433	.5017	.4395
CombSUM	△.2962	△.6395	▲.6973	▲.6482	▲.5513
ClustFuseCombSUM	△.2951	△.6385	△.6927	▲.6407	▲.5476
BurstFuseCombSUM	△.3047	△.6408	▲.6983	▲.6497	▲.5613
CombMNZ	△.2948	△.6350	△.6918	△.6347	△.5420
ClustFuseCombMNZ	.2886	.6312	.6873	.6283	.5416
BurstFuseCombMNZ	△.3027	△.6398	▲.6972	▲.6443	▲.5524
λ-Merge	.2847	.6275	.6876	.6279	.5383
BurstFuseλ-Merge	.2942	.6387	.6894	.6326	▽.5428
TimeRA	**▲.3117**▲△	**▲.6518**△△	**▲.7023**△	**▲.6545**▲	**▲.5733**▲△

when compared against BurstFuseX, traditional and state-of-the-art, supervised and unsupervised data fusion approaches for microblog post search. Additional analyses show that TimeRA is a robust and efficient data fusion method that outperforms state-of-the-art temporal retrieval algorithms.

As to future work, we plan to look into combining social information, such as user relationships into data fusion and further analyze our model in scenarios where the documents being searched are published in bursts. In addition, matrix factorization is not only one way to inferring scores of missing rating in collaborative filtering. Other ways for score inferences include, for instance, latent semantic models (Hofmann, 2004). In the future, we plan to apply alternative collaborative filtering technologies to infer scores of missing documents in data fusion. In the next chapter, we will turn to explore data fusion strategies for ad hoc search.

5.A Derivation of The Models

The following is the derivation of (5.10). We leave out the derivations of (5.2), (5.3), (5.4), (5.5) and (5.6) as they may be obtained in a similar way.

$$\begin{aligned}
\frac{\partial C_2}{\partial \mathbf{S}_i} =&(1-\beta)\sum_{j=1}^{n} I_{ij} w(d_j|L_i)(g(\mathbf{S}_i^\top \mathbf{V}_j) - R_{ij}) g'(\mathbf{S}_i^\top \mathbf{V}_j)\mathbf{V}_j \\
&+ \sum_{j=1}^{n}\sum_{b\in\mathcal{B}} \frac{\beta}{|\{d_k : d_k L_i d_j|b\}|} \sum_{\substack{d_k\in b \\ d_k L_i d_j}} I_{ij} I_{ik} r(d_j|d_k) \\
&\cdot w(d_k|L_i)(g(\mathbf{S}_i^\top \mathbf{V}_j) - R_{ik}) g'(\mathbf{S}_i^\top \mathbf{V}_j)\mathbf{V}_j + \lambda_1 \mathbf{S}_i^\top \\
=&(1-\beta)\sum_{i=1}^{m} I_{ij} w(d_j|L_i)(g(\mathbf{S}_i^\top \mathbf{V}_j) - R_{ij}) g'(\mathbf{S}_i^\top \mathbf{V}_j)\mathbf{S}_i^\top \\
&+ \sum_{i=1}^{m}\sum_{b\in\mathcal{B}} \frac{\beta}{|\{d_k : d_k L_i d_j|b\}|} \sum_{\substack{d_k\in b \\ d_k L_i d_j}} I_{ij} I_{ik} r(d_j|d_k) \\
&\cdot w(d_k|L_i)(g(\mathbf{S}_i^\top \mathbf{V}_j) - R_{ik}) g'(\mathbf{S}_i^\top \mathbf{V}_j)\mathbf{S}_i^\top + \lambda_2 \mathbf{V}_j,
\end{aligned} \tag{5.13}$$

where $g'(x) = \exp(x)/(1+\exp(x))^2$ is the derivative of the logistic function $g(x) = 1/(1+\exp(-x))$.

6
Manifold-based Data Fusion

In the previous two chapters, Chapters 4 and 5, we have explored data fusion approaches for microblog search, where the approaches mainly focus on utilizing the ranks of the documents in the result lists. In the experiments on which we report in Chapters 4 and 5, the cluster-based data fusion methods are not able to beat our proposed fusion approaches, although cluster-based data fusion methods leverage the content of microblog posts. Our proposed methods do not utilize the content of the posts, in fact, the experiments showed that using the content of the posts make the retrieval performance become a little worse and time-consuming. On Twitter people use abbreviations or change their spelling in creative ways so as to fit their message in the allotted space, giving rise to a rather idiomatic language and resulting in the fact that content-based fusion may not work well.

In contrast, in the setting of ad hoc search, the content of documents is more reliable and useful for retrieval models compared to those in microblogging platforms. In this chapter, we focus on data fusion strategies that combine not only the ranks of the documents in the result lists but also the content of the documents for ad hoc search. State-of-the-art content-based fusion approaches, i.e., cluster-based fusion approaches, have demonstrated that integrating the content of documents into data fusion can improve the performance of data fusion and thus improve the performance of ad hoc search. Cluster-based fusion approaches first cluster documents appearing in the lists to be fused and then allow for a document appearing low in a single list to be promoted if it is similar to other highly ranked documents in the cluster (Kozorovitsky and Kurland, 2011). While intuitive, a non-relevant document should not be "promoted" even if it is in the cluster that contains a large number of high ranked documents. We propose a richer structure than afforded by clusters, namely manifolds (Thurston and Milnor, 1979) (see Section 2.3.4). The research question we address in this chapter is the following:

RQ 3 Can manifolds be used to improve data fusion performance for ad hoc search?

To answer the question, we propose to use a richer structure to infer (possible) relevance of a document contained in a ranked list to be merged: manifolds. Specifically, we propose a novel manifold-based data fusion approach, *ManX*, that integrates manifolds constructed by using inter-document similarities and ranking information produced by a standard fusion method *X* (such as CombSUM or CombMNZ). Our fusion approach first constructs manifolds for documents in the lists to be merged, then takes the top k

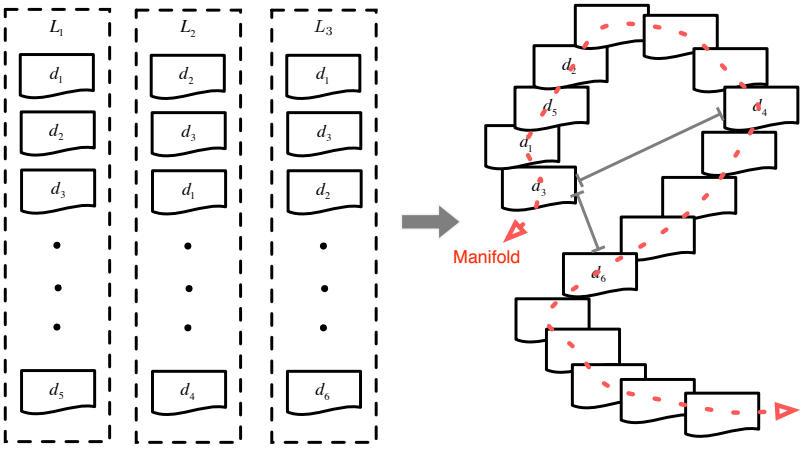


Figure 6.1: Rewarding a document that is similar to (assumed) relevant documents and whose manifold distance to those documents is short. In response to a given query, three lists need to be fused. Documents d_5, d_4 and d_6 appear only once in the lists and ranked low, while documents d_1, d_2 and d_3 appear multiple times in the lists and highly ranked. In terms of the relevances of d_4 and d_6, d_4 is more likely to be relevant. Because the total manifold distances from d_4 to d_1, d_2 and d_3 are small than those between d_6 and d_1, d_2 and d_3, although the total Euclidean distances are opposite. Note the manifold distance between two documents is the length of the trail from one document to another (shown in red dashed line in the figure); and the Euclidean distance between two documents is the point-to-point distance between one document to another (shown in gray solid lines in the figure).

retrieved documents in the merged list produced by the standard fusion method *X* as pseudo relevant documents. Subsequently, it boosts documents that are similar to the pseudo relevant documents in the manifold: if a document d is similar to (pseudo) relevant documents d_1, d_2 ..., d_i and the manifold distances between d and d_1, d_2 ..., d_i are short, then d can be "rewarded" and is also assumed to be relevant.

Fig. 6.1 illustrates the idea of our manifold-based data fusion approach. There, three lists need to be fused. Because d_1, d_2 and d_3 are ranked very high in almost all of the lists and the manifold distances between d_5 and them are quite short, d_5 can be "promoted" to being relevant, although d_5 may appear only in list L_1 and is ranked very low in this list. According to the cluster-based fusion method (Kozorovitsky and Kurland, 2011), the point to point distance (Wang et al., 2008) between d_6 and the relevant document d_3 is shorter than that between d_4 and d_3, d_6 and d_3 are in the same cluster, and then d_6 is more likely to be relevant than d_4. This is not true in the example. In fact, d_4 is more likely to be relevant than d_6, as its manifold distance to d_3 (following the red trail; see Fig. 6.1) is shorter than d_6 even if d_4 is not in the cluster of d_3.

Furthermore, previous work on data fusion mainly focuses on enhancing the retrieval performance of data fusion and ignores the running time. In fact, there are some bot-

tlenecks associated with state-of-the-art data fusion algorithms. For instance, in (Kozorovitsky and Kurland, 2011) document similarity scores need to be obtained for each document pair and the time complexity is expensive namely the method needs to compute similarity scores for $n \times (n-1)/2$ document pairs before processing data fusion, which is computationally prohibitive in the situation that large number of documents need to fused. To enhance the fusion efficiency, we propose to use top-k documents in the fusion list generated by some standard fusion method as anchors.

In our experiments aimed at assessing the performance of our manifold-based data fusion method, ManX, we sample runs that have been submitted to TREC-3, TREC-10 and TREC-12 and fuse them using our proposed method. For the fusion method X on top of which ManX builds, we consider two unsupervised fusion approaches: CombSUM and CombMNZ. We also use a recent state-of-the-art method: cluster-based data fusion as one of our baselines. Experimental results demonstrate that ManX significantly outperforms all of the baseline methods and runs faster than state-of-the-art fusion methods.

Our contributions in this chapter can be summarized as:

i. We propose a data fusion method for ad hoc document search, ManX, which not only takes traditional information such as retrieval status value or rank into account, but also exploits manifold structure of the documents in the lists.
ii. To the best of our knowledge, this is the first attempt to utilize manifold strategies for data fusion in IR.
iii. As far as we are aware, ours is also the first attempt to use anchor documents in data fusion for efficiency.

The remainder of the chapter is organized as follows: Section 6.1 provides an analysis of the cluster-based fusion methods; Section 6.2 details our proposed manifold-based data fusion method; Section 6.3 describes our experimental setup; Section 6.4 presents our experimental results; finally, Section 6.5 concludes this chapter.

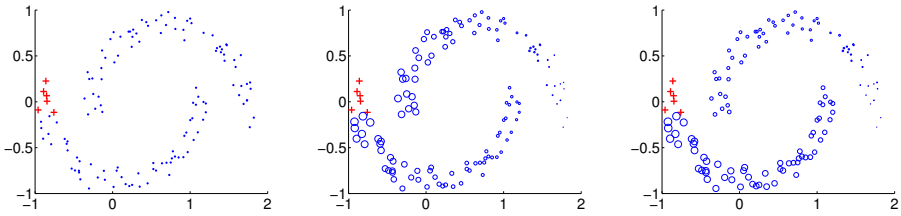
6.1 Analysis of Cluster-Based Fusion

In Section 2.3.1 we described cluster-based fusion methods, which we abbreviate as ClustFuseX for convenient discussion in this chapter. Here we briefly analyze such methods. The cluster hypothesis has two assumptions (Cao et al., 2013; Liu et al., 2010, 2012b; Zhou et al., 2004):

i Similar documents are likely to have similar relevance to the same information need.
ii Documents in the same structure (cluster, manifold, etc.) are likely to have similar relevance to the same information need.

The first assumption imposes a constraint on similar documents, i.e., documents located close to each other in some similarity space. Therefore, this assumption is local. The second assumption talks about documents within the same structure (e.g., a cluster), which may be spread across the similarity space and, thus, this assumption is global.

Many clustering techniques, including the nearest-neighbor-based approach used by cluster-based data fusion methods, ClustFuseX, depend only on the local assumption



(a) Distribution of the documents in $\mathbb{R}^2$.

(b) Ranking generated by a nearest-neighbor-based technique.

(c) Ideal ranking generated by a manifold-base technique.

Figure 6.2: Distribution of example documents in $\mathbb{R}^2$ and different rankings for these documents. Relevant documents are shown in the lower moon, whereas non-relevant documents are shown in the upper moon. The red markers '+' indicate the top-ranked documents. The remaining documents are marked with blue circles 'o'. The size of the blue markers is proportional to the rank of corresponding documents. Here, the x-axis and y-axis are the intervals of the distribution of the example data.

and fail in cases where both assumptions are required (Zhou et al., 2004). To illustrate this, consider the example in Figure 6.2. Here, relevant documents are shown in the lower moon and non-relevant documents are shown in the upper moon. The top-ranked relevant documents are indicated by red markers '+', while other documents are marked with blue circles 'o', where the size of a circle is proportional to the rank of the corresponding document. As can be seen in Figure 6.2(b), if we rank documents using a cluster-based retrieval method, many non-relevant documents, that are close to the red crosses, will be ranked higher than relevant documents which are further away. However, as can be seen in Figure 6.2(c), if we rank documents using ideal manifold-based retrieval, all relevant documents can be ranked higher than any of non-relevant documents.

Another drawback of ClustFuseX is its high computational cost. ClustFuseX performs the following steps to compute fusion scores: (i) estimates a language model of each document in $\mathcal{C}_\mathcal{L}$, (ii) computes inter-document similarities, (iii) creates clusters for all $d \in \mathcal{C}_\mathcal{L}$, (iv) computes fusion scores $f_{\text{ClustFuseX}}(d; q)$, and (v) ranks documents based on their scores. For simplicity, we treat basic operations (summation, multiplication, etc.) with equal importance. Under this simplifying assumption, the complexity of ClustFuseX steps is the following:

i Estimating the language model of a document requires $O(avg\_dl)$ time, where avg_dl is the average document length in $\mathcal{C}_\mathcal{L}$. Thus, the first step of ClustFuseX has complexity of $O(avg\_dl \cdot |\mathcal{C}_\mathcal{L}|)$.

ii Computing the similarity $\text{sim}(d_1, d_2)$ requires $O(avg\_vl)$ time, where avg_vl is the average vocabulary length of a document in $\mathcal{C}_\mathcal{L}$. Therefore, computing inter-document similarities for all document pairs in $\mathcal{C}_\mathcal{L}$ requires $O(avg\_vl \cdot |\mathcal{C}_\mathcal{L}|^2)$ time.

iii Creating a cluster of size δ for each document in $\mathcal{C}_\mathcal{L}$ requires $O(|\mathcal{C}_\mathcal{L}| \cdot \log(\delta))$ time.

iv Calculating ClustFuseX scores $f_{\text{ClustFuseX}}(d; q)$ has $O(|\mathcal{C}_\mathcal{L}|^2)$ complexity.

v Reranking documents based on $f_{\text{ClustFuseX}}(d;q)$ requires $O(|\mathcal{C}_{\mathcal{L}}|)$ time.

Summing up the estimated cost of each step and dropping minor summands, we get the following time complexity of ClustFuseX: $O(avg\_dl \cdot |\mathcal{C}_{\mathcal{L}}| + avg\_vl \cdot |\mathcal{C}_{\mathcal{L}}|^2)$. According to the Heap's law, the vocabulary size of a document is sublinear with regards to the document length. Still these two quantities are comparable and, therefore, we assume that $avg\_dl < avg\_vl \cdot |\mathcal{C}_{\mathcal{L}}|$. Under this assumption, the time complexity of ClustFuseX is $O(avg\_vl \cdot |\mathcal{C}_{\mathcal{L}}|^2)$, which can be very large for documents with a rich vocabulary and large $\mathcal{C}_{\mathcal{L}}$.

In this chapter we aim to address the drawbacks of ClustFuseX by developing an efficient data fusion technique, which utilizes inter-document similarities and considers both assumptions, underlying the clustering hypothesis.

6.2 Manifold-Based Data Fusion

In this section we propose a novel data fusion technique, ManX, which is built on the assumption that if a document d is similar to other relevant documents in a manifold, then d can be promoted in response to a query. ManX makes use of both assumptions underlying the cluster hypothesis and, therefore, we expect it to be at least as effective as the ClustFuseX method, which builds on a similar idea. Moreover, we develop an efficient version of ManX, improving the time complexity compared to ClustFuseX, which makes our method applicable to real-work data fusion tasks.

6.2.1 Optimization problem

Recall that we are given a set of ranked lists $\mathbf{L} = \{L_1, \ldots, L_m\}$, returned in response to a query q by m retrieval systems. Our goal is to calculate a fusion score $f(d;q)$ for each document $d \in \mathcal{C}_{\mathbf{L}}$, where $\mathcal{C}_{\mathbf{L}} := \bigcup_{i=1}^{m} L_i$, and then rank these documents by their fusion scores to form a single result list L_f.

Our goal is to consider the inter-document similarities in $\mathcal{C}_{\mathbf{L}}$ and to promote documents that are similar to that are known or assumed to be relevant documents. To achieve this goal, we will regularize fusion scores $f_{\text{X}}(d;q)$, produced by a data fusion method X, using the document similarity information. In particular, we define a matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{C}_{\mathbf{L}}| \times |\mathcal{C}_{\mathbf{L}}|}$ of inter-document similarities, where $w_{ij} = \text{sim}(d_i, d_j)$ for all pairs of documents in $\mathcal{C}_{\mathbf{L}}$. Then we compute a regularized score $f_{\text{ManX}}(d;q)$ for each document $d \in \mathcal{C}_{\mathbf{L}}$ by solving the following optimization problem:

$$\min_{\mathbf{f}_{\text{ManX}}} \mathcal{Q}(\mathbf{f}_{\text{ManX}}) = \frac{1}{2}||\mathbf{f}_{\text{ManX}} - \mathbf{f}_{\text{X}}||^2 + \frac{\eta}{2}\text{tr}(\mathbf{f}_{\text{ManX}}^{\top}\mathbf{M}\mathbf{f}_{\text{ManX}}), \tag{6.1}$$

where $\mathbf{f}_{\text{X}}$ is a set of fusion scores produced by a method X: $\mathbf{f}_{\text{X}} = [f_{\text{X}}(d_1;q), \ldots, f_{\text{X}}(d_{\mathcal{C}_{\mathbf{L}}};q)]^{\top}$, $\mathbf{f}_{\text{ManX}}$ is a set of regularized fusion scores, which need to be found, η is a regularization parameter and $\mathbf{M}$ is the laplacian matrix: $\mathbf{M} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D} = \text{diag}(D_{11}, \ldots, D_{|\mathcal{C}_{\mathbf{L}}||\mathcal{C}_{\mathbf{L}}|})$ is a diagonal matrix defined by $D_{ii} = \sum_{j=1}^{|\mathcal{C}_{\mathbf{L}}|} W_{ij}$.

The first component in (6.1) forces the ManX fusion scores $\mathbf{f}_{\text{ManX}}$ to be close to the original fusion scores $\mathbf{f}_{\text{X}}$. The second component smoothes $\mathbf{f}_{\text{X}}$ by assigning similar

scores to similar documents. The amount of smoothing is controlled by the parameter η. The final fused list L_f is constructed by ranking documents $d \in \mathcal{C}_{\mathbf{L}}$ according to their regularized fusion scores $f_{\mathrm{ManX}}(d; q)$.

Based on the two assumptions indicated in the cluster hypothesis, we expect our proposed manifold-based fusion method work better than cluster-based fusion method. Theoretically, we work with manifold distances among the documents to be fused rather than the point-to-point distances; that is for a document, unlike cluster-based fusion where only the distances between this document and the other documents in the same cluster are consider, we consider all the distances between it and all other documents to be fused.

6.2.2 Optimal solution

To solve the optimization problem in (6.1), we need to take a derivative of $\mathcal{Q}(\mathbf{f}_{\mathrm{ManX}})$ and equate it to 0:

$$\frac{\partial \mathcal{Q}(\mathbf{f}_{\mathrm{ManX}})}{\partial \mathbf{f}_{\mathrm{ManX}}} = \mathbf{f}_{\mathrm{ManX}} - \mathbf{S}\mathbf{f}_{\mathrm{ManX}} + \eta(\mathbf{f}_{\mathrm{ManX}} - \mathbf{f}_{\mathrm{X}}) = 0, \tag{6.2}$$

where $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$. Letting $\alpha = 1/(1+\eta)$, we can obtain the following closed-form solution for $\mathbf{f}_{\mathrm{ManX}}$:

$$\mathbf{f}_{\mathrm{ManX}} = (1-\alpha)(\mathbf{I} - \alpha\mathbf{S})^{-1}\mathbf{f}_{\mathrm{X}}, \tag{6.3}$$

where $\mathbf{I}$ is the identity matrix.

If the set of documents $\mathcal{C}_{\mathbf{L}}$ is large, then the computation of $(\mathbf{I} - \alpha\mathbf{S})^{-1}$ is intractable. In this case an iterative algorithm, like HITS (Kleinberg, 1999), can be used to solve (6.2). Let $\mathbf{f}_{\mathrm{ManX}}(t)$ denote a set of regularized fusion scores at a t-th iteration. Also, let $\mathbf{f}_{\mathrm{ManX}}(0) = \mathbf{f}_{\mathbf{X}}$. Then the regularized fusion scores on the next iteration can be calculated as follows:

$$\mathbf{f}_{\mathrm{ManX}}(t+1) = \alpha\mathbf{S}\mathbf{f}_{\mathrm{ManX}}(t) + (1-\alpha)\mathbf{f}_{\mathbf{X}}. \tag{6.4}$$

This process is repeated until $\mathbf{f}_{\mathrm{ManX}}(t+1)$ converges. After convergence, with (6.4) we can easily obtain the final score $\mathbf{f}_{\mathrm{ManX}}(t+1)$ as the solution to (6.1) as the follows:

$$\mathbf{f}_{\mathrm{ManX}}(t+1) = (\alpha\mathbf{S})^t\mathbf{f}_{\mathbf{X}} + (1-\alpha)\sum_{i=0}^{t}(\alpha\mathbf{S})^i\mathbf{f}_{\mathrm{X}}. \tag{6.5}$$

6.2.3 Efficient ManX

The ManX data fusion technique, as presented in Section 6.2.1, has the same efficiency issue as the cluster-based approach: the pairwise distances between all documents in $\mathcal{C}_{\mathbf{L}}$ need to be computed, which requires $O(avg\_vl \cdot |\mathcal{C}_{\mathbf{L}}|^2)$ time. Since data fusion has to run on the fly and promptly present fused results to users, this problem has to be solved for techniques, like ClustFuseX and ManX, to be applicable in practice. In the rest of this section we will focus on addressing the above issue and develop an efficient version of ManX.

Defining Anchors

In a variety of real world information retrieval applications, including web search, online advertising, users mainly pay attention to the top-k documents and ignore other documents ranked low (Culpepper et al., 2012; Hofmann et al., 2013). Following this idea, we will consider the top-k documents ($k \ll |\mathcal{C}_\mathbf{L}|$), produced by a basic data fusion method X, and treat them as anchors for other documents in $\mathcal{C}_\mathbf{L}$. In other words, the definition of anchors in our setting is that: anchors are the documents that are ranked within top-k by a standard data fusion method, e.g., CombSUM.

Recall that ManX uses a data fusion method X to obtain initial fusion scores $\mathbf{f}_\mathrm{X} = [f_\mathrm{X}(d_1; q), \ldots, f_\mathrm{X}(d_{\mathcal{C}_\mathbf{L}}; q)]^\top$. The top-$k$ documents in this list are assumed to be relevant to the query q. We will denote their fusion scores as $\mathbf{a}_\mathrm{X} = [f_\mathrm{X}(d_1; q), \ldots, f_\mathrm{X}(d_k; q)]^\top$. Then, according to Liu et al. (2010), we represent other regularized fusion scores as a linear combination of scores in $\mathbf{a}_\mathrm{ManX}$:

$$f_\mathrm{ManX}(d_i; q) = \sum_{j=1}^{k} Z_{ij} f_\mathrm{ManX}(a_j; q), \tag{6.6}$$

where Z_{ij} are the weights discussed below. In matrix form this can be written as follows:

$$\mathbf{f}_\mathrm{ManX} = \mathbf{Z}\mathbf{a}_\mathrm{ManX}. \tag{6.7}$$

According to Liu et al. (2010), a good design principle for the weight matrix $\mathbf{Z}$ is to have $\sum_{j=1}^{k} Z_{ij} = 1$ and $Z_{ij} \geq 0$. Therefore, we define Z_{ij} as follows:

$$Z_{ij} = \frac{W_{ij}}{\sum_{l=1}^{k} W_{il}}, \tag{6.8}$$

where $W_{ij} = \mathrm{sim}(d_i, a_j)$. This means that the more similar a document d_i and an anchor a_j, the higher the weight Z_{ij}. Thus, documents similar to anchors will have higher regularized scores, which is a desired property, because we assume anchor-documents to be relevant.

Efficient Optimal Solution

Let us define the similarity matrix $\mathbf{W}$ as follows:

$$\mathbf{W} = \mathbf{Z}\mathbf{Z}^T. \tag{6.9}$$

According to this definition, two documents d_i and d_j have a certain similarity $w_{ij} > 0$ if they share at least one anchor-document (anchor-document is the document that is ranked within top-k by a standard fusion method) d_l: $z_{il} \neq 0 \;\&\; z_{jl} \neq 0$. The more anchors are shared, the more similar are the documents to each other. Note that, according to the definition of $\mathbf{Z}$, the similarity matrix $\mathbf{W}$ is non-negative and, thus, the laplacian $\mathbf{M} = \mathbf{D} - \mathbf{W}$ is positive semidefinite (Chung, 1997).

According to (6.7), fusion scores of documents in $\mathcal{C}_\mathbf{L}$ can be represented as a linear combination of fusion scores of anchor-documents: $\mathbf{f}_\mathrm{ManX} = \mathbf{Z}\mathbf{a}_\mathrm{ManX}$. Therefore, $\mathbf{f}_\mathrm{ManX}$ in (6.3) can be substitute with $\mathbf{Z}\mathbf{a}_\mathrm{ManX}$ as follows:

$$\mathbf{f}^*_\mathrm{ManX} = \mathbf{Z}\mathbf{a}_\mathrm{ManX} = (1-\alpha)(\mathbf{I} - \alpha\mathbf{S})^{-1}\mathbf{f}_\mathrm{X}, \tag{6.10}$$

where $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}\mathbf{Z}\mathbf{Z}^{\top}\mathbf{D}^{-1/2}$. According to (6.9), i.e., $\mathbf{W} = \mathbf{Z}\mathbf{Z}^{\top}$, and Let $\mathbf{P} = \mathbf{Z}^{\top}\mathbf{D}^{-\frac{1}{2}}$, (6.10) comes:

$$\begin{aligned}\mathbf{f}^{*} = \mathbf{Z}\mathbf{a}_{\mathrm{ManX}} &= (1-\alpha)(\mathbf{I}_n - \alpha\mathbf{D}^{-\frac{1}{2}}\mathbf{Z}\mathbf{Z}^{T}\mathbf{D}^{-\frac{1}{2}})^{-1}\mathbf{f}_{\mathrm{X}} \\ &= (1-\alpha)(\mathbf{I}_n - \mathbf{P}^{\top}(\mathbf{P}\mathbf{P}^{\top} - \frac{1}{\alpha}\mathbf{I}_k)^{-1}\mathbf{P})\mathbf{f}_{\mathrm{X}},\end{aligned} \tag{6.11}$$

By multiplying $(\mathbf{I}_n - \alpha\mathbf{D}^{-\frac{1}{2}}\mathbf{Z}\mathbf{Z}^{T}\mathbf{D}^{-\frac{1}{2}})$ and $(\mathbf{I}_n - \mathbf{P}^{\top}(\mathbf{P}\mathbf{P}^{\top} - \frac{1}{\alpha}\mathbf{I}_k)^{-1}\mathbf{P})$ we obtain an identity matrix, thereby proving (6.11). Here is the proof.

Proof. We have:

$$\begin{aligned}&(\mathbf{I}_n - \alpha\mathbf{D}^{-\frac{1}{2}}\mathbf{Z}\mathbf{Z}^{\top}\mathbf{D}^{-\frac{1}{2}}) \times (\mathbf{I}_n - \mathbf{P}^{\top}(\mathbf{P}\mathbf{P}^{\top} - \frac{1}{\alpha}\mathbf{I}_k)^{-1}\mathbf{P}) \\ &= (\mathbf{I}_n - \alpha\mathbf{P}^{\top}\mathbf{P}) \times (\mathbf{I}_n - \mathbf{P}^{\top}(\mathbf{P}\mathbf{P}^{\top} - \frac{1}{\alpha}\mathbf{I}_k)^{-1}\mathbf{P}) \\ &= \mathbf{I}_n - \mathbf{P}^{\top}(\mathbf{P}\mathbf{P}^{\top} - \frac{1}{\alpha}\mathbf{I}_k)^{-1}\mathbf{P} - \alpha\mathbf{P}^{\top}\mathbf{P} + \alpha\mathbf{P}^{\top}\mathbf{P}\mathbf{P}^{\top}(\mathbf{P}\mathbf{P}^{\top} - \frac{1}{\alpha})^{-1}\mathbf{P} \\ &= \mathbf{I}_n - (\mathbf{P}^{\top} - \alpha\mathbf{P}^{\top}\mathbf{P}\mathbf{P}^{\top})(\mathbf{P}\mathbf{P}^{\top} - \frac{1}{\alpha}\mathbf{I}_k)^{-1}\mathbf{P} - \alpha\mathbf{P}^{\top}\mathbf{P} \\ &= \mathbf{I}_n - \alpha\mathbf{P}^{\top}(\frac{1}{\alpha}\mathbf{I}_k - \mathbf{P}\mathbf{P}^{\top})(\mathbf{P}\mathbf{P}^{\top} - \frac{1}{\alpha}\mathbf{I}_k)^{-1}\mathbf{P} - \alpha\mathbf{P}^{\top}\mathbf{P} \\ &= \mathbf{I}_n + \alpha\mathbf{P}^{\top}\mathbf{P} - \alpha\mathbf{P}^{\top}\mathbf{P} \\ &= \mathbf{I}_n\end{aligned}$$

□

In the first transition, we replace $\mathbf{S}$ with $\mathbf{P}^{\top}\mathbf{P}$. Then we open the brackets and perform the multiplication. In the third transition, we group the summands containing $(\mathbf{P}\mathbf{P}^{\top} - \frac{1}{\alpha}\mathbf{I}_k)^{-1}$. Then we move $\mathbf{P}^{\top}$ out, so that matrix $(\frac{1}{\alpha}\mathbf{I}_k - \mathbf{P}\mathbf{P}^{\top})$ cancels with its inverse. The remaining matrix $\alpha\mathbf{P}^{\top}\mathbf{P}$ also cancels and we get the identity matrix $\mathbf{I}_n$ as a result. This proves that (6.10) and (6.11) are indeed equivalent.

6.2.4 Analysis of efficient ManX

There are two important properties in (6.11). The expensive computational cost of inverting an $n \times n$ matrix can be reduced to inverting a $k \times k$ matrix, where $k \ll n$. This reduction in computational cost is able to significantly speed up the fusion of merging rank lists. Thus, the proposed manifold based fusion method can merge the rank lists fast and still tries to boost the fusion performance. In addition, during the computation process, as indicated by (6.9), we only need to save and use the smaller matrix $\mathbf{Z}$ rather than the adjacency matrix $\mathbf{W}$ in memory, which allows our efficient manifold based fusion method to be applied to very large data sets or memory-short environments.

Efficient ManX performs the following steps:

i Similarly to ClustFuse, ManX estimates language models of all documents in $\mathcal{C}_{\mathbf{L}}$, which requires $O(avg\_dl \cdot n)$ time, where avg_dl is the average document length in $\mathcal{C}_{\mathbf{L}}$ and $n = |\mathcal{C}_{\mathbf{L}}|$ (see Section 6.1 for more details).

ii Calculate similarities between the top-k documents and other documents in $\mathcal{C}_{\mathbf{L}}$, which takes $O(avg\_vl \cdot kn)$ time, where avg_vl is the average vocabulary length of documents in $\mathcal{C}_{\mathbf{L}}$.

iii Using (6.8), construct matrix $\mathbf{Z}$. This requires $O(kn)$ time, because the denominator in (6.8) is computed n times.

iv Using (6.9), construct matrix $\mathbf{W}$. This takes $O(kn^2)$ time due to matrix multiplication.

v Similarly, the construction of matrices $\mathbf{D}$ and $\mathbf{P}$ requires $O(n)$ and $O(kn)$ time respectively.

vi Invert matrix $\mathbf{P}\mathbf{P}^\top - \frac{1}{\alpha}\mathbf{I}_k$ in $O(k^3)$ time.

vii Calculate fusion scores according to (6.11), which additionally takes $O(k^2n+kn^2) = O(kn^2)$ time for matrix multiplication.

Summing up the above times and eliminating minor terms, we get that the overall time complexity of ManX is $O(avg\_vl \cdot kn + kn^2)$. In Section 6.1 we showed that the complexity of ClustFuse is $O(avg\_vl \cdot n^2)$. Since k is a small number, ManX should be more efficient than ClustFuse. We will verify this theoretical result experimentally in Section 6.4.2.

6.3 Experimental Setup

In this section, we describe our experimental setup. In particular, we list our research questions in Section 6.3.1, describe our baselines in Section 6.3.2, and details the experiments in Section 6.3.3.

6.3.1 Detailed research questions

To answer the main research question (RQ 3) in this chapter, we divide it into several research questions. The research questions guiding the remainder of the chapter are:

RQ 3.1 Does ManX outperform the best run to be fused, the standard data fusion method that it incorporates, and cluster-based data fusion methods? (See Section 6.4.1 for the answer.)

RQ 3.2 What is the effect of the number of lists to be fused in ManX? (See Section 6.4.2 for the answer.)

RQ 3.3 How does the number of top-k documents used as anchor documents impact the performance of ManX? (See Section 6.4.3 for the answer.)

RQ 3.4 Does ManX run faster than state-of-the-art data fusion methods? (See Section 6.4.4 for the answer.)

RQ 3.5 Can we actually observe the hypothesized effect sketched in Fig. 6.1? (See Section 6.4.5 for the answer.)

RQ 3.6 Can LDA rather than tokens of documents be utilized to compute document similarities for ManSUM? (See Section 6.4.6 for the answer.)

In order to answer our research questions we work with 3 text collections and the corresponding submitted runs provided by the ad hoc track of TREC-3 (Harman, 1994), the web track of TREC-10 (Hawking and Craswell, 2002) and the robust retrieval track of TREC-12 (Voorhees, 2005), respectively (see Section 3.2 for details).

6.3.2 Baselines and evaluation

We compare the proposed ManX (using the top 20 documents as anchor documents) to 4 fusion baselines: 2 traditional fusion methods that ManX builds on, i.e., CombSUM and CombMNZ, 2 state-of-the-art cluster-based fusion methods, ClustFuseCombSUM and ClustFuseCombMNZ. As ManX utilizes top-k documents as anchor documents, we also compare ManX to another version of ManX, viz., ManX2. ManX2 is the same as ManX except that ManX2 utilizes all documents as anchor documents for data fusion. For cluster-based fusion, our ManX and ManX2 methods, we use a 35/10/5 split of the dataset for our training, validation and test sets, respectively. We train the fusion model by varying values of parameters. The best values of the parameters are then chosen on the validation set, and evaluated on the test queries. The train/validation/test splits are permuted until all test queries were chosen once for the test set. We repeat the experiments 10 times and report the average evaluation results.

For building an index to the data set in our experiments, we use the Lemur toolkit[1] and apply tokenization, Porter stemming, and remove stop words using the INQUERY list. For computing the similarity between two documents in ManX and the baseline methods, we first apply LDA to capture multinomial distribution of topics specific to all the documents in $\mathcal{C}_{\mathcal{L}}$ and then compute the similarity score via symmetric Kullback-Leibler divergence based on the documents' multinomial distribution of topics. For evaluation we use trec_eval[2] to evaluate the MAP, p@5, p@10, p@20 nDCG@5, nDCG@10 and nDCG@20 performance for our methods and baselines. See Section 3.3.1 for details about the metrics we use in this chapter.

In the following, we let ManSUM, ManMNZ, ManSUM2, ManMNZ2, ClustSUM, ClustMNZ be short for ManFuseCombSUM, ManFuseCombMNZ, ManFuseCombSUM2, ManFuseCombMNZ2, ClustFuseCombSUM and ClustFuseCombMNZ, respectively.

6.3.3 Experiments

We report 6 main experiments aimed at answering the research questions listed in Section 6.3.1. Our first experiment aims at understanding the overall performance of ManX, we choose 5 best runs out of the runs produced by the participants in the TREC-3 ad hoc track, the TREC-10 web track and the TREC-12 robust retrieval track based on the runs' p@20 distribution. See Table 6.1 for a summary of the top 5 best runs in each edition of TREC.

[1] http://www.lemurproject.org
[2] http://trec.nist.gov

Table 6.1: Summary of top 5 runs in each track used for the experiments.

TREC	Top 5 runs	Performance
TREC-3	inq102, citya1, brkly7, citya2, assctv2	$0.6110 \leq$ p@20 < 0.6740
TREC-10	iit01m, csiro0mwa1, ok10wtnd1, flabxtd, ok10wtnd0	$0.3590 \leq$ p@20 < 0.4730
TREC-12	pircRBa2, pircRBa1, pircRBd2, pircRBd3, aplrob03a	$0.3640 \leq$ p@20 ≤ 0.3930

Then, to understand the effect of the number of lists to be merged, we randomly choose $m = 3, 5, \ldots, 25$ lists from the 40 lists in TREC-3 and fuse the lists using ManX, the standard fusion methods, and the cluster-based method. For each m, we repeat the experiments 20 times and we report the average results and standard deviation. Next, to understand the effect of the number of top-k documents used as anchor documents in ManX, we vary the number of top-k documents from $k = 5, 10, \ldots, 30$ in our experiments and look into the results. In order to understand the efficiency of ManX, we again randomly fuse $m = 3, 5, 9, 15$ and 23 lists from the TREC-3 dataset for all the 50 queries and report the average time required. To observe the hypothesized effect sketched in Fig. 6.1, we provide a query-level performance analysis between ManX and ClustFuseX. Finally, we examine the effectiveness of utilizing LDA rather than tokens of documents in computing document similarities in ManX.

6.4 Results and Analysis

In Section 6.4.1 we show the results of fusing the top 5 lists; Section 6.4.2 analyzes the number of lists on the overall performance; Section 6.4.3 shows the effect of the number of anchor documents in ManX; Section 6.4.4 makes run time comparisons between ManX and the baselines. Finally, Section 6.4.5 provides a topic-level analysis.

6.4.1 Fusing the top component runs

The performance of ManX, ClustFuseX and the standard fusion method X they incorporate is presented in Table 6.2, with the runs from the 3 editions of TREC mentioned in Section 6.3.3. From Table 6.2, we can see that in terms of any metric, MAP, p@5, p@10, p@20, nDCG@5, nDCG@10 and nDCG@20, ManX, ClustFuseX and the standard fusion methods they integrate can perform substantially better than any of the top 5 runs in each TREC dataset in most cases. Most of the improvements are statistically significant.

Additionally, it is worth noting in Table 6.2 that ManX outperforms ClustFuseX for all the 3 datasets where in almost all the cases such improvements are statistically significant. This finding shows that utilizing the two prior assumptions of both local and global consistency in the cluster hypothesis for data fusion can help to enhance the retrieval performance. Interestingly, although ManX utilizes only the top 20 documents as anchor documents, it is not statistically significantly worse than ManX2 where all the documents are taken as anchor documents. ManX performs a litter better than ManX2 in some cases.

For instance, when fusing the top 5 runs from TREC-12, ManSUM outperforms ManSUM2 a little bit. This illustrates that we can adapt anchor graph in data fusion to reduce the time complexity but still maintain the effectiveness of manifold based data fusion.

In the following, we use ManSUM, ClustSUM and CombSUM as representative examples for making comparisons as the results of other methods are qualitatively similar to those comparisons among ManSUM, ClustSUM and CombSUM.

6.4.2 Effect of the number of lists being merged

We have already seen that ManX outperforms cluster-based data fusion and the standard fusion methods when fusing top 5 runs in different tracks. We now turn to explore the effect on the performance of ManX of varying the number of lists to be merged. Fig. 6.3, we randomly choose $m \in \{3, 5, \ldots, 25\}$ lists from the 40 lists submitted to the ad hoc track in TREC-3. Here we only report results using the TREC 3 runs; the findings using other runs from other edition of the TREC tracks are qualitatively similar. For each m, we repeat the experiment 20 times and report on the average scores and the corresponding standard deviations. In each experiment on each metric, we record the best performance score denoted as "Maximum" and the average performance score of the input runs denoted as "Average."

As shown in Fig. 6.3, ManSUM outperforms ClustSUM in terms of all the metrics and the performance gaps remain almost unchanged, in absolute terms, no matter how many component runs are fused. One reason for this is that ManSUM inherits the merits of cluster-based fusion methods as it fully utilizes both local and global consistency in the cluster hypothesis for data fusion, whereas ClustSUM only considers the local consistency. Additionally, it is clear from Fig. 6.3 that in most cases on most metrics, both ManSUM and ClustSUM outperform the best input run "Maximum" no matter how many runs are fused. This illustrates that documents that are ranked low in the input runs but that are construed in the same manifolds (clusters) where a large number of top ranked documents are can be rewarded to be relevant. All the fusion methods, CombSUM, ClustSUM and ManSUM can easily beat the average performance of the input runs.

6.4.3 Effect of anchor documents

Next we examine the effect of using different numbers of top-k documents in our effective manifold based fusion method. How does the number of top-k documents used as anchor documents impact the performance of ManX? We again use ManSUM as a representative example. Fig. 6.4 depicts the MAP, p@k and nDCG@k performance curves for ManSUM, ClustSUM and CombSUM when fusing the top 5 runs from the ad hoc track in TREC-3. As the number of top documents used as anchor documents in manifold based fusion method, ManSUM, increases, the performance become better. This is because more anchor documents are included, more relevant documents are used to boost the performance in ManSUM. The performance improvement seems to level off when more than 20 top documents are used as anchor documents. This is perhaps because as more documents are included, more relevant documents but also more non-relevant are within the top-k documents. The performance of ManSUM can beat ClustSUM when

Table 6.2: Retrieval performance on the top 5 best lists from the TREC-3, TREC-10 and TREC-10 tracks. The best performing run per metric per track is in boldface. Statistically significant differences between ManX and ClustFuseX, and between ManX and ManX2, are marked in the upper left hand corner of the ManX score, and the upper right hand corner of the ManX score, respectively.

			p@			nDCG@		
		MAP	5	10	20	5	10	20
TREC-3	assctv2	.0959	.7280	.6760	.6110	.7540	.7089	.6533
	brkly7	.0949	.7600	.7120	.6490	.7582	.7257	.6777
	citya1	.0932	.7400	.7120	.6640	.7486	.7265	.6874
	citya2	.0876	.7320	.6940	.6280	.7255	.7016	.6533
	inq102	.1039	.7440	.7220	.6740	.7423	.7275	.6925
	CombSUM	.1073	.8040	.7620	.6960	.8009	.7736	.7245
	CluSUM	.1199	.8200	.8020	.7430	.8136	.8038	.7638
	ManSUM	▲.1312	▲.8440	▲.8120▽	▲.7840▽	▲.8313	▲.8136	▲.7958
	ManSUM2	.1317	**.8480**	**.8260**	**.7960**	**.8336**	**.8238**	**.8054**
	CombMNZ	.1065	.8080	.7700	.6970	.8021	.7781	.7254
	CluMNZ	.1236	.8240	.8040	.7630	.8127	.8031	.7766
	ManMNZ	▲.1305	▲.8400	▲.8200	▲.7910	▲.8260	▲.8173	▲.8001
	ManMNZ2	**.1324**	.8240	.8240	.7930	.8148	.8183	.8004
TREC-10	csiro0mwa1	.1729	.5440	.5020	.4170	.4688	.4768	.4583
	flabxtd	.1184	.5000	.4460	.3660	.4396	.4263	.4038
	iit01m	.2145	.6320	.5880	.4730	.5650	.5707	.5369
	ok10wtnd0	.1265	.5360	.4540	.3590	.4575	.4456	.4105
	ok10wtnd1	.1408	.5560	.4680	.4010	.4766	.4529	.4408
	CombSUM	.1988	.6480	.5620	.4650	.5870	.5679	.5409
	CluSUM	.2351	.6400	.5980	.5370	.5821	.5945	.5955
	ManSUM	▲.2742	▲.6880	▲.6600	▲**.6140**	▲.6078▽	▲.6310	▲.6496
	ManSUM2	.2734	.7040	.6580	.6020	.6293	.6369	.6447
	CombMNZ	.1999	.6640	.5640	.4560	.6024	.5751	.5381
	CluMNZ	.2392	.6760	.6020	.5390	.6125	.6043	.5996
	ManMNZ	▲**.2858**△	▲.6960▽	▲.6580▼	▲.6100	▲.6298	▲.6457	▲**.6554**
	ManMNZ2	.2748	**.7160**	**.6800**	.5990	**.6365**	**.6540**	.6474
TREC-12	aplrob03a	.1774	.5140	.4510	.3640	.4643	.4409	.4241
	pircRBa1	.1843	.5200	.4540	.3890	.4832	.4572	.4470
	pircRBa2	.1849	.5280	.4880	.3930	.5004	.4892	.4580
	pircRBd2	.1733	.5260	.4570	.3705	.4778	.4565	.4307
	pircRBd3	.1745	.5220	.4490	.3665	.4828	.4542	.4319
	CombSUM	.1899	.5480	.4870	.3980	.5082	.4866	.4632
	CluSUM	.2213	.5720	.5420	.4655	.5225	.5258	.5197
	ManSUM	▲.2545	▲.6300▽	▲.6000	▲.5145	▲.5657	▲.5709	▲.5652
	ManSUM2	.2498	.6140	.5910	.5085	.5527	.5637	.5611
	CombMNZ	.1914	.5540	.4900	.4025	.5121	.4883	.4673
	CluMNZ	.2331	.6100	.5590	.4775	.5503	.5427	.5358
	ManMNZ	▲**.2583**	▲**.6400**△	▲**.6190**	▲**.5165**	▲**.5752**△	▲**.5884**△	▲**.5727**△
	ManMNZ2	.2507	.6260	.5980	.5075	.5628	.5718	.5622

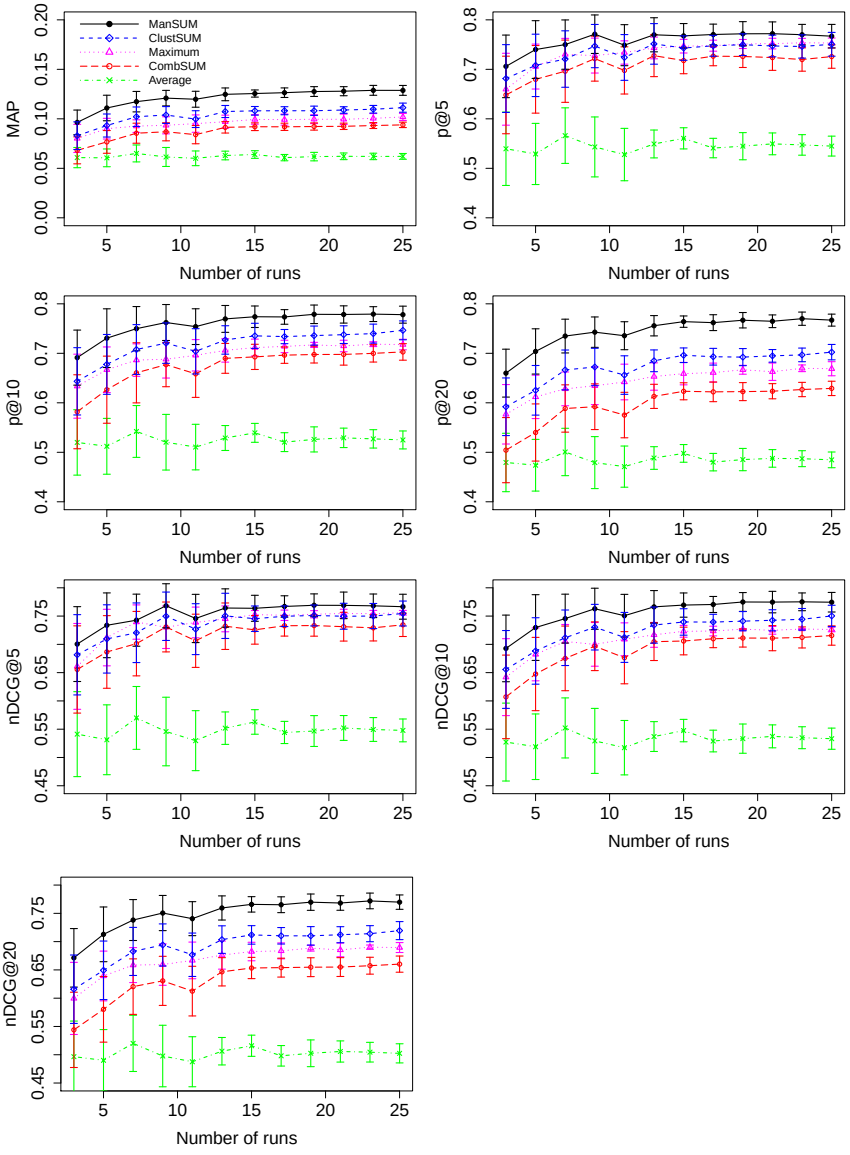


Figure 6.3: Effect on performance (in terms of MAP, p@5, p@10, p@20, nDCG@5, nDCG@10 and nDCG@20) of the number of component runs, using runs randomly sampled from the TREC-3 ad hoc track. We plot averages and standard deviations. Note: the figures are not to the same scale.

Table 6.3: Time spent on fusing runs by different fusion methods. Recorded in seconds with standard deviations (std).

	Number of lists									
	3		5		9		15		23	
CombSUM	3.98e–4	(3.28e–5)	8.07e–4	(1.13e–4)	1.69e–3	(3.88e–4)	2.86e–3	(4.61e–4)	3.96e–3	(6.07e–4)
ClustSUM	3.59	(6.39e–1)	11.27	(2.53)	42.91	(5.09)	117.45	(10.97)	267.36	(18.02)
ManSUM	1.46e–1	(4.65e–2)	5.09e–1	(8.26e–2)	1.79	(4.25e–1)	4.48	(6.64e–1)	11.17	(2.19)

as few as 10 top document are used as anchor documents in most cases. This illustrates that taking only a small number of top-k documents as anchor documents can get better performance.

6.4.4 Run time comparisons

We now explore how fast ManX can merge component runs in response to a query. Again, we take ManSUM, ClustSUM and CombSUM as representative examples. These fusion methods are developed in C++ and the experiments are run on a 10.9.2 MacBook Pro computer with 4GB memory and a 2.3 GHz Intel core i5 processor. In Table 6.3, we randomly choose $m = 3$, 5, 9, 15 and 23 runs from the 40 runs submitted to the ad hoc track in TREC 3. For each m, we repeat the experiment 20 times and report the average run time per query (in seconds) that the fusion methods require.

As can be seen in Table 6.3, ManSUM does not run as fast as the standard fusion method it integrates, CombSUM, but it manages to merge runs fast. ManSUM merges the runs with 11.17s when given 23 result runs and with 0.15s when fusing 3 runs. As the number of runs to be fused increases, the time spent on fusing is almost linear for CombSUM and ManSUM. For ClustSUM, however, the time increases dramatically with larger numbers of component runs. When fusing 23 runs, ClustSUM needs to spend 267.36s, although it only spends 3.59s on fusing 3 result runs. Combining the findings in Fig. 6.4 and Table 6.3, we can conclude that manifold-based fusion methods not only runs faster than cluster based-fusion ones but also achieves better retrieval performance.

6.4.5 Topic-level analysis

We take a closer look at per query improvements of ManX over ClustFuseX and the underlying standard fusion method X. For brevity, we only consider ManSUM as a representative example and report query performance differences against ClustSUM which outperforms CombSUM. We only consider fusing runs submitted to the ad hoc track in TREC-3 as other comparison results are quantitively similar.

Fig. 6.5 shows the per query performance differences in terms of MAP, p@k and nDCG@k, respectively, of ManSUM against ClustSUM. ManSUM achieves performance improvements for many queries when compared against ClustSUM, and many of the differences are relatively big. This shows that in many cases fusion based on manifold distances is to be preferred over fusion over point-to-point similarities as they are used in ClustFuseX. For instance, in response to query #190 in TREC-3, ClustSUM ranks the relevant document #ZF109-569-003 at the eighth position, whereas ManSUM works better

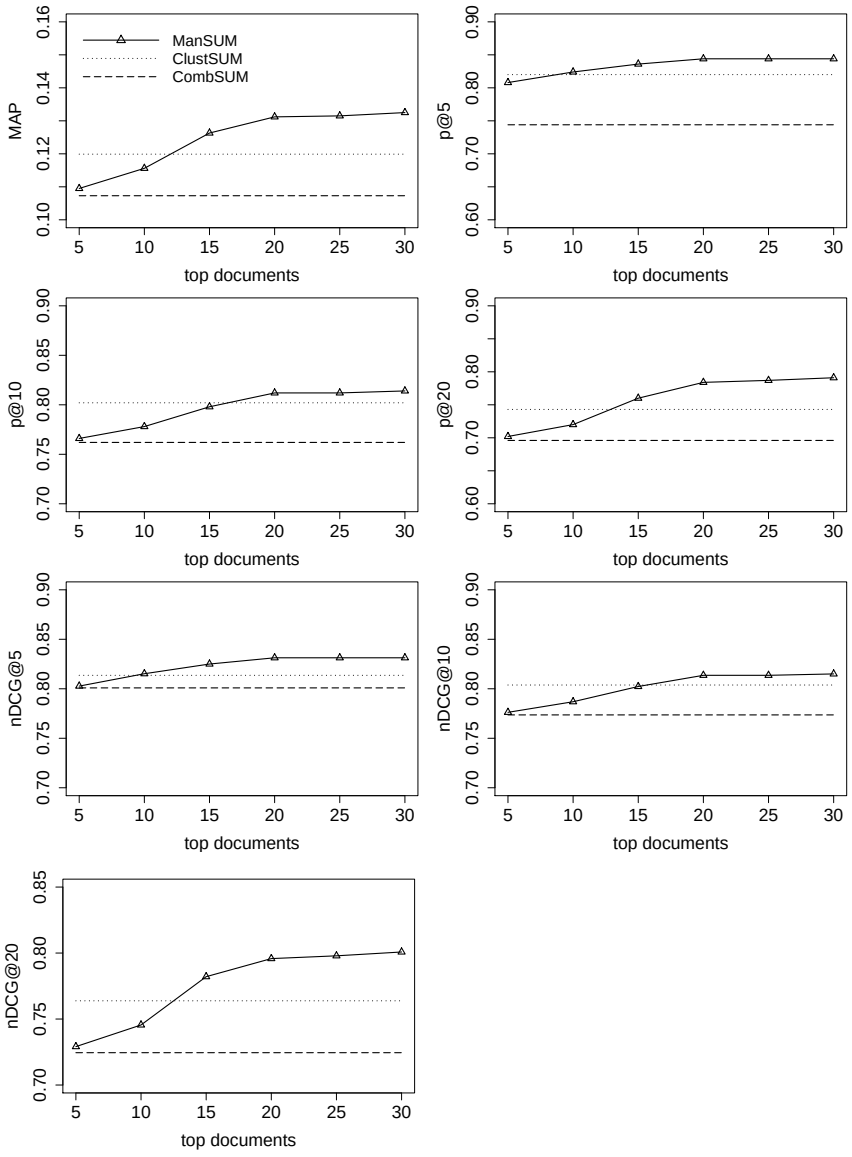


Figure 6.4: Effect on performance (in terms of MAP, p@5, p@10, p@20, nDCG@5, nDCG@10 and nDCG@20) of the number of anchor documents being utilized when fusing the top 5 runs from the ad hoc track in TREC-3. Note: the figures are not to the same scale.

Table 6.4: Fusion performance of ManSUM and ManSUM-token. ManSUM-token is the same as ManSUM except that it computes document-to-document similarity scores based on tokens of the documents rather than LDA. Statistically significant differences between ManSUM and ManSUM-token are marked in the upper left hand corner of the ManSUM score.

		p@			nDCG@		
	MAP	5	10	20	5	10	20
ManSUM-token	.1263	.8360	.7980	.7680	.8250	.8023	.7821
ManSUM	.1312	.8440	.8120	$^{\triangle}$.7840	.8313	$^{\triangle}$.8136	$^{\triangle}$.7958

and ranks the document at the fifth position. In contrast, in a very small number of cases, ManSUM performs poorer than ClustSUM. This appears to be due to the fact that some non-relevant documents in manifolds where there are a number of relevant documents ranked highly in the result runs are promoted by ManSUM. For instance, in response to query #176 in TREC-3, ClusteSUM ranks the non-relevant document #AP880609-0187 at the tenth position, but ManSUM becomes worse and ranks it at the sixth position.

6.4.6 Document similarity

Finally, we examine the effect on the overall performance of using LDA to compute document-to-document similarities used in ManX, and contrast the performance of ManX-token using tokens directly to compute the similarities. Here, ManX-token is the same algorithm as ManX except that for computing the similarities it utilizes tokens of documents directly rather than multinomial distribution of topics specific to the documents. We again use ManSUM and ManSUM-token as representative examples only.

Table 6.4 shows the experimental results. As we can see in the table, although the performance of ManSUM and ManSUM-token is not statistically significant different on the metrics MAP, p@5, p@10 and nDCG@5, the performance of ManSUM on p@20, nDCG@10 and nDCG@20 statistically significantly outperforms that of ManSUM-token. This is because LDA can capture the topic similarities between documents rather than just the token level similarities. In addition, the time complexity of utilizing LDA to obtain the similarities is smaller than that of utilizing tokens of documents. The finding in Table 6.4 plus the analysis of the time complexity show that utilizing LDA in ManSUM not only helps to fuse result lists faster but also improve the performance.

6.5 Conclusion

In this chapter, we have introduced a novel data fusion approach, ManX, which is based on manifolds to merge ranked lists that are retrieved in response to a given query. In ManX, manifolds of documents in the lists to be fused are constructed by utilizing inter-document similarities that are computed based on multinomial distributions of topics specific to documents. ManX fully utilizes two prior assumptions in the cluster hypoth-

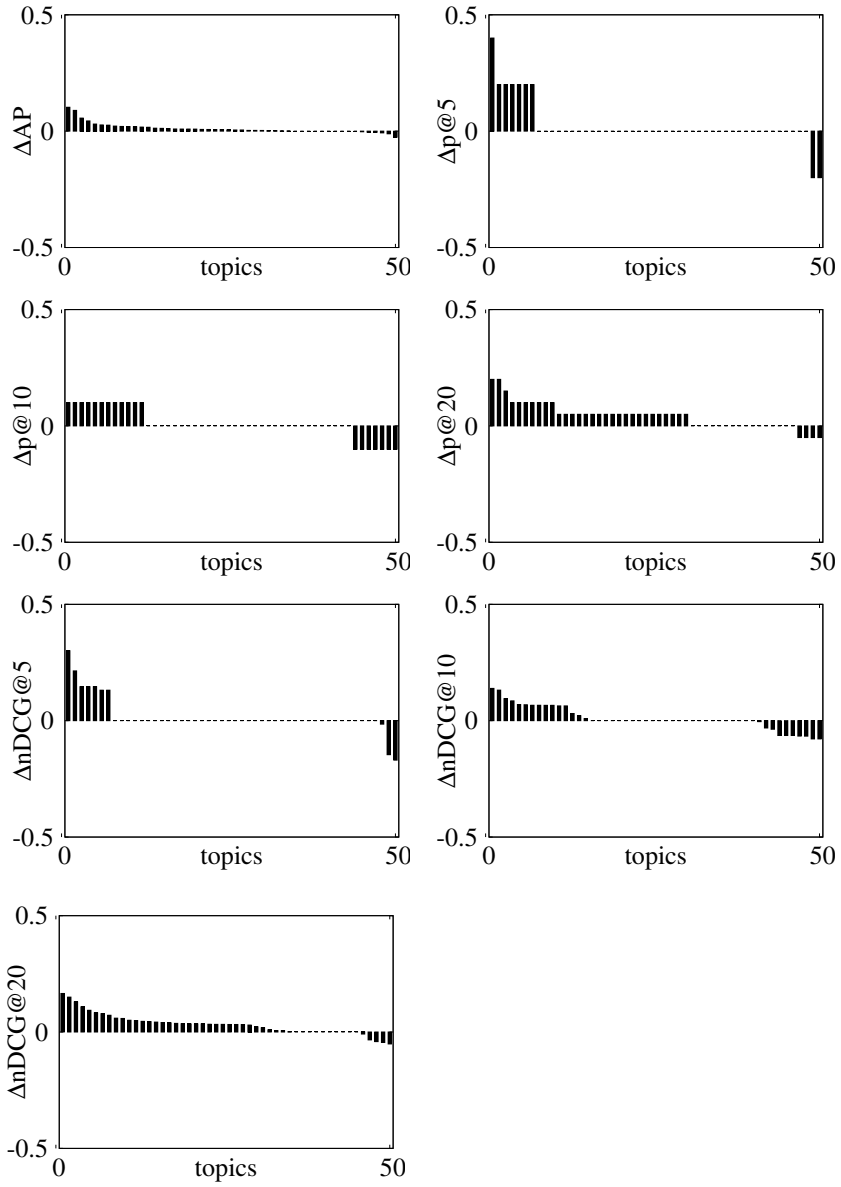


Figure 6.5: Per query performance difference of ManSUM against ClustSUM. The figures shown are for fusing the runs in TREC-3 ad hoc track., for MAP, p@5, p@10, p@20, nDCG@5, nDCG@10 and nDCG@20 (from left to right, top to bottom). A bar extending above the center of a plot indicates that ManSUM outperforms ClustSUM, and vice versa for bars below the center.

esis, thereby enabling it to reward documents that are ranked low in only few lists but surrounding which there are many highly relevant documents in the same manifold. Furthermore, it takes the top-k documents from the fused list merged with a standard fusion method X as anchor documents to make fusion process become faster. Our experiments provide an answer to the main research question raised at the beginning of this chapter:

RQ 3 Can manifolds be used to improve data fusion performance for ad hoc search?

To answer the main research question, we worked with 3 text collections provided by the TREC-3 ad hoc, TREC-10 web and TREC-12 robust retrieval tracks. We made comparisons between our ManX and the baseline fusion methods. Our experimental results demonstrated that richer structure of documents can be used to improve fusion performance, and ManX not only outperforms the standard fusion methods that it integrates and a state-of-the-art data fusion method that leverages clustering strategies, but it also fuses result lists more efficiently.

As to future work, we envisage to combine more useful information to improve the construction of manifolds for data fusion. We believe that utilizing semantic linking techniques (Meij et al., 2012), i.e., first detecting entities in documents, and then expanding the documents with other external material such as the Wikipedia articles the entities link to, would help to improve the quality of manifold such that the fusion methods can improve the performance. In this chapter we have use top-k documents as anchors to improve the efficiency of running manifold-based data fusion methods. Another direction for the future work is to consider other more effective strategies for selecting anchors.

In Chapters 4 and 5 we focused on data fusion for searching posts in microblogging environments. In this chapter we mainly focused on how to utilize manifolds of documents in the result lists to improve the fusion performance. In the next chapter (Chapter 7) we will investigate whether data fusion can improve the performance of search result diversification.

7

Fusion Helps Diversification

We have shown in Chapters 4 and 5 that data fusion can improve the performance of microblog search, and in Chapter 6 we have shown the performance improvements of ad hoc search due to data fusion. In this chapter we continue our study on data fusion, but in a different setting and with a different application, i.e., search result diversification. Search result diversification is widely being studied as a way of tackling query ambiguity. Instead of trying to identify the "correct" interpretation behind a query, the idea is to make the search results diversified so that users with different backgrounds and intents will find at least one of these results to be relevant to their information need (Agrawal et al., 2009). In contrast to the traditional assumption of independent document relevance, search result diversification approaches typically consider the relevance of a document in light of other retrieved documents (Santos et al., 2011). Diversification models try to identify the probable "aspects" of the query and return documents for each aspect, thereby making the result list more diverse.

As we have seen in Chapters 4, 5 and 6, data fusion methods can improve retrieval performance in terms of traditional relevance-oriented metrics like MAP and precision@k over the methods used to generate the individual result lists being fused. One reason is that retrieval approaches often return very different non-relevant documents, but many of the same relevant documents (Wu, 2012). In this chapter we explore whether data fusion can improve performance in terms of diversity metrics. Thus, we are interested in answering:

RQ 4 Can data fusion help search result diversification?

We examine the hypothesis that *data fusion can improve performance in terms of diversity metrics by promoting aspects that are found in disparate ranked lists to the top of the fused list*. Our first step in testing this hypothesis is to examine the impact of existing data fusion methods in terms of diversity scores when fusing ranked lists. We find that they tend to improve over individual component runs on nearly all of the diversity metrics that we consider: Prec-IA, MAP-IA, α-NDCG, ERR-IA (all at rank 20).

Building on these findings we propose a new data fusion method, called diversified data fusion (DDF). Based on latent Dirichlet allocation (LDA), it operates on documents in the result lists to be fused, whether the result lists have been diversified or not. DDF infers latent topics, their probabilities of being relevant and a multinomial distribution of

topics over the documents being fused. Thus, it integrates topic structure and rank information. DDF does not assume the explicit availability of query aspects, but infers these as well as the latent prior for a given query via the documents being fused. Experimental results show that DDF can aggregate result lists—whether produced by diversification or ad hoc retrieval models—and boost the diversity of the final fused list, outperforming state-of-the-art diversification methods and established data fusion methods, especially in terms of intent-aware precision metrics.

Our contributions in this chapter can be summarized as follows:

i. We tackle the challenge of search result diversification in a novel way by using data fusion methods.

ii. We propose a novel data fusion method that aims at optimizing diversification measures and that proves to be especially effective in terms of intent-aware precision metrics.

iii. We analyze the effectiveness of data fusion for result diversification and find that our fusion method as well as other fusion methods can significantly outperform state-of-the-art diversification methods.

The remainder of the chapter is organized as follows. Section 7.1 describes our proposed diversified data fusion model for search result diversification. Section 7.2 describes our experimental setup. Section 7.3 is devoted to our experimental results and we conclude in Section 7.4.

7.1 Diversified Data Fusion

We first summarize the main notation used in this chapter in Table 7.1; other additional notation used in this thesis can be referred to Table 2.1. In the remainder of this chapter, we distinguish between queries, aspects and topics. A *query* is an expression of an information need; in our experimental evaluation below, queries are provided as part of a TREC test collection. An *aspect* (sometimes called subtopic at the TREC Web track) is an interpretation of an information need. We use *topic* to refer to latent topics as identified by a topic modeling method, in our case LDA. A *component list* is a ranked list that serves as input for a data fusion method. A *fused* list is a list that is the result of applying a fusion method to component lists. A natural and direct way of diversifying a result list in the setting of data fusion is this: first rank the documents in the component lists by their estimated relevance to the query through a standard data fusion method, such as CombSUM, and then diversify the ranking through effective search result diversification models, such as MMR (Carbonell and Goldstein, 1998) and PM-2 (Dang and Croft, 2012). In our experiments, we implement two more baselines, called CombSUM-MMR and CombSUMPM-2. They first use CombSUM to obtain a fused list and then use MMR and PM-2, respectively, to diversify the list. See Section 2.3.1 for how to compute CombSUM for each documents in the result lists.

We propose a diversified data fusion (DDF) method that not only inherits the merits of traditional data fusion methods, i.e., it can improve the performance on relevance

Table 7.1: Additional notation used in the chapter (cf. Table 2.1).

Notation	Gloss
z	topic
N_d	number of tokens in d
$\mathcal{R}$	set of top ranked documents
$qt[z\|q]$	quotient score for z given q in PM-2 algorithm Dang and Croft (2012)
$v_{z\|q}$	probability of z given q
$s_{z\|q}$	"portion" of seat occupied by z given q in PM-2
λ	a free trade-off parameter in PM-2
α	the parameter of a topic Dirichlet prior
β	the parameter of a token Dirichlet prior
T	number of topics
V	number of unique tokens in $\mathcal{C}_{\mathbf{L}}$
θ_d	multinomial distribution of topics specific to d
ϕ_z	multinomial distribution of tokens specific to topic z
μ_z	mean of Log-normal distribution of fusion scores for topic z
σ_z	deviation of Log-normal distribution of fusion scores for z
z_{di}	topic associated with the i-th token in the document d
w_{di}	i-th token in document d
f_{di}	fusion score for token w_{di}

orientated metrics, but also considers a query as a compound rather than a single representation of an underlying information need, and regards documents appearing in the component lists as mixtures of latent topics.

Overview of DDF

DDF consists of three main parts: (I) perform standard data fusion; (II) infer latent topics; (III) perform diversification; see Algorithm 3. In the first part ("Part I" in Algorithm 3), DDF computes the fusion scores of the documents in the component lists based on an existing unsupervised data fusion method (steps 2 and 3 in Algorithm 3); in this chapter we use CombSUM, as our experimental results in Section 7.3.1 and Section 7.3.2 show that CombSUM outperforms other plain fusion methods in most cases. In the second part ("Part II" in Algorithm 3), DDF integrates fusion scores into an LDA topic model such that latent topics of the documents, their corresponding estimated relevance scores, and the multinomial distribution of the topics specific to each document can be inferred (steps 5–17 in Algorithm 3). In the last part ("Part III" in Algorithm 3), DDF uses the outputs of Parts I and II as input for an existing diversification method; in this chapter, we use PM-2 (Dang and Croft, 2012) because it is a the state-of-the-art search result diversification model. Some concepts in PM-2, such as "quotient" and "seat," play important roles in the definition of the diversification step; they have been discussed in Section 2.3.5.

Below we describe how to infer latent topics ("Part II" in Algorithm 3) and how we utilize the information generated from latent topics and fusion scores ("Part III").

Algorithm 3: Diversified data fusion

Input : A query q
Ranked lists to be fused, $L_1, L_2, \ldots, L_m$
The combined set of documents $\mathcal{C}_{\mathbf{L}} := \bigcup_{i=1}^{m} L_i$
A standard fusion method X
A tradeoff parameter λ
Number of latent topics T
Hyperparameters α, β

Output: A final fused diversified list of documents L_f.

/* Part I: Perform standard data fusion */
for $d = 1, 2, \ldots, |\mathcal{C}_{\mathbf{L}}|$ **do**
  Initialize $f_{\mathrm{X}}(d|\mathbf{L}, q)$ using a standard fusion method X
/* Part II: Infer latent topics */
Randomly initialize topic assignment for all tokens in $\mathbf{w}$
for $z = 1, 2, \ldots, T$ **do**
  Initialize μ_z and σ_z randomly for topic z
for $iter = 1, 2, \ldots, N_{iter}$ **do**
  for $d = 1, 2, \ldots, |\mathcal{C}_{\mathbf{L}}|$ **do**
    for $i = 1, 2, \ldots, N_d$ **do**
      draw z_{di} from $P(z_{di}|\mathbf{w}, \mathbf{r}, \mathbf{z}_{-di}, \alpha, \beta, \mu, \sigma, \mathbf{L}, q)$
      update $n_{z_{di} w_{di}}$ and $m_{d z_{di}}$
  for $z = 1, 2, \ldots, T$ **do**
    update μ_z and σ_z
Compute the posterior estimate of θ
for $z = 1, 2, \ldots, T$ **do**
  $v_{z|q} \leftarrow \frac{\exp\{u_z + \frac{1}{2}\sigma_z^2\}}{\sum_{z'=1}^{T} \exp\{u_{z'} + \frac{1}{2}\sigma_{z'}^2\}}$
/* Part III: Perform diversification */
$L_f \leftarrow \varnothing$
$\mathcal{R} \leftarrow \mathcal{C}_{\mathbf{L}}$
for $z = 1, 2, \ldots, T$ **do**
  $s_{z|q} \leftarrow 0$
for *all positions in the ranked list* L_f **do**
  for $z = 1, 2, \ldots, T$ **do**
    $qt[z|q] = \frac{v_{z|q}}{2 s_{z|q} + 1}$
  $z^* \leftarrow \arg\max_z qt[z|q]$
  $d^* \leftarrow \arg\max_{d \in \mathcal{R}} \ \lambda \times qt[z^*|q] \times P(d|z^*, q) +$
    $(1 - \lambda) \sum_{z \neq z^*} qt[z|q] \times P(d|z, q)$
  $L_f \leftarrow L_f \cup \{d^*\}$ /* append d^* to L_f */
  $\mathcal{R} \leftarrow \mathcal{R} \backslash \{d^*\}$
  for $z = 1, 2, \ldots, T$ **do**
    $s_{z|q} \leftarrow s_{z|q} + \frac{P(d^*|z,q)}{\sum_{z'} P(d^*|z',q)}$

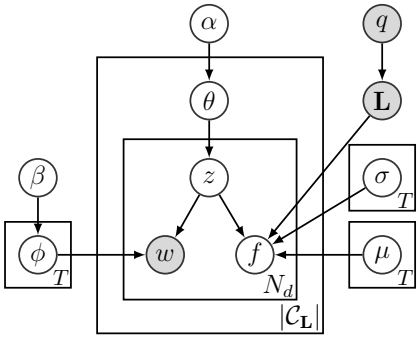


Figure 7.1: DDF graphical model for Gibbs sampling.

Part II: Inferring latent topics

Previous work on search result diversification shows that explicitly computing the probabilities of aspects of a query can improve diversification performance (Abbar et al., 2013; He et al., 2012; Santos et al., 2010a). We do not assume that aspect information is explicitly available; we infer latent topics and their probabilities of being relevant using topic modeling.

Topic discovery in DDF is influenced not only by token co-occurrences, but also by the fusion scores of documents in the component lists. To avoid normalization and because fusion scores of the documents theoretically belong to $(0, +\infty)$, we employ a log-normal distribution for fusion scores to infer latent topics of the query via the documents and their relevance probabilities.

The latent topic model used in DDF is a generative model of relevance and the tokens in the documents that appear in the component individual lists, i.e., documents in $\mathcal{C}_{\mathbf{L}}$. The generative process used in Gibbs sampling (Liu, 1994) for parameter estimation, is as follows:

i. Draw T multinomials ϕ_z from a Dirichlet prior β, one for each topic z;

ii. For each document $d \in \mathcal{C}_{\mathbf{L}}$, draw a multinomial θ_d from a Dirichlet prior α; then for each token w_{di} in document d:

   (a) Draw a topic z_{di} from multinomial θ_d;

   (b) Draw a token w_{di} from multinomial $\phi_{z_{di}}$;

   (c) Draw a fusion score f_{di} for w_{di} from Log-normal $\mathcal{N}(\mu_{z_{di}}, \sigma_{z_{di}})$.

Fig. 7.1 shows a graphical representation of our model. In the generative process, the fusion scores of tokens observed in the same document are the same and computed by a data fusion method, like CombSUM, for the document, although a fusion score is generated for each token from the log-normal distribution. We use a fixed number of latent topics, T, although a non-parametric Bayes version of DDF that automatically integrates over the number of topics would certainly be possible. The posterior distribution of topics depends on the information from two modalities—both tokens and the fusion scores of the documents.

Inference is intractable in this model. Following (Griffiths and Steyvers, 2004; Liu, 1994; Ren et al., 2013; Wang and McCallum, 2006; Wei and Croft, 2006; Xu et al., 2012), we employ Gibbs sampling to perform approximate inference. We adopt a conjugate prior (Dirichlet) for the multinomial distributions, and thus we can easily integrate out θ and ϕ, analytically capturing the uncertainty associated with them. In this way we facilitate the sampling, i.e., we need not sample θ and ϕ at all. Because we use the continuous log-normal distribution rather than discretizing fusion scores, sparsity is not a big concern in fitting the model. For simplicity and speed we estimate these log-normal distributions μ and σ by the method of moments, once per iteration of Gibbs sampling (see the Appendix 7.A). We find that the sensitivity of the hyper-parameters α and β is limited. Thus, for simplicity, we use fixed symmetric Dirichlet distributions ($\alpha = 50/T$ and $\beta = 0.1$) in all our experiments.

In the Gibbs sampling procedure above, we need to calculate the conditional distribution $P(z_{di}|\mathbf{w}, \mathbf{r}, \mathbf{z}_{-di}, \alpha, \beta, \mu, \sigma, \mathbf{L}, q)$ (step 11 in Algorithm 3), where $\mathbf{z}_{-di}$ represents the topic assignments for all tokens except w_{di}. We begin with the joint probability of documents to be fused, and using the chain rule, we can obtain the conditional probability conveniently as:

$$
\begin{aligned}
&P(z_{di}|\mathbf{w}, \mathbf{r}, \mathbf{z}_{-di}, \alpha, \beta, \mu, \sigma, \mathbf{L}, q) \propto \\
&\quad (m_{dz_{di}} + \alpha_{z_{di}} - 1) \times \frac{n_{z_{di}w_{di}} + \beta_{w_{di}} - 1}{\sum_{v=1}^{V}(n_{z_{di}v} + \beta_v) - 1} \times \\
&\quad \frac{1}{f_{\mathrm{X}}(d|\mathbf{L}, q)\sigma_{z_{di}}\sqrt{2\pi}} \exp\{-\frac{(\ln f_{\mathrm{X}}(d|\mathbf{L}, q) - \mu_{z_{di}})^2}{2\sigma_{z_{di}}^2}\},
\end{aligned}
$$

where n_{zv} is the total number of tokens v that are assigned to topic z, m_{dz} represents the number of tokens in document d that are assigned to topic z. An overview of the Gibbs sampling procedure we use is shown from step 5 to step 14 in Algorithm 3; details are provided in the Appendix 7.A.

One merit of our generative model for DDF is that we can predict a fusion score for any document once the tokens in the document have been observed. Given a document, we predict its fusion score by choosing the discretized fusion score that maximizes the posterior which is calculated by multiplying the fusion score probability of all tokens from their corresponding topic-wise log-normal distributions. Then the fusion score for a document d can be obtained by:

$$
f_X(d|\mathbf{L}, q) = \arg\max_f \prod_{i=1}^{N_d} p(f|\mu_{z_i}, \sigma_{z_i}).
$$

More importantly, after the Gibbs sampling procedure, we can easily infer the multinomial distribution of topics specific to each document $d \in \mathcal{C}_{\mathbf{L}}$ as (step 15 in Algorithm 3):

$$
\theta_{d,z} = \frac{n_{d,z} + \alpha_z}{\sum_{z=1}^{T}(n_{d,z} + \alpha_z)}, \tag{7.1}
$$

where $n_{d,z}$ is the number of tokens assigned to latent topic z in document d; we can also conveniently estimate the probability of a topic being relevant to the query, denoted as

$v_{z|q}$, by (step 17 in Algorithm 3):

$$v_{z|q} := \frac{\mathbb{E}[\mathbf{f}|z]}{\sum_{z'=1}^{T}\mathbb{E}[\mathbf{f}|z']} = \frac{\exp\{u_z + \frac{1}{2}\sigma_z^2\}}{\sum_{z'=1}^{T}\exp\{u_{z'} + \frac{1}{2}\sigma_{z'}^2\}}, \tag{7.2}$$

where $\mathbb{E}$ denotes the expectation.

Part III: Diversification

In Part III of our DDF model we propose a modification of PM-2. Before we discuss the details of this modification, we briefly describe PM-2. See Section 2.3.5 for more details of PM-2. PM-2 is a probabilistic adaptation of the Sainte-Laguë method for assigning seats (positions in the fused list) to members of competing political parties (aspects) such that the number of seats for each party is proportional to the votes (aspect popularity, also called aspect probabilities, i.e., $p(z|q)$) they receive. PM-2 starts with a ranked list L_f with k empty seats. For each of these seats, it computes the quotient $qt[z|q]$ for each topic z given q following the Sainte-Laguë formula:

$$qt[z|q] = \frac{v_{z|q}}{2s_{z|q}+1}, \tag{7.3}$$

where $v_{z|q}$ is the probability of topic z given q, i.e., the weight of topic z. According to the Sainte-Laguë method, this seat should be awarded to the topic with the largest quotient in order to best maintain the proportionality of the list. Therefore, PM-2 assigns the current seat to the topic z^* with the largest quotient. The document to fill this seat is the one that is not only relevant to z^* but to other topics as well:

$$\begin{aligned} d^* \quad = \quad & \arg\max_{d\in\mathcal{R}} \big(\lambda \times qt[z^*|q] \times P(d|z^*,q) + \\ & (1-\lambda)\textstyle\sum_{z\neq z^*} qt[z|q] \times P(d|z,q)\big), \end{aligned} \tag{7.4}$$

where $P(d|z,q)$ is the probability of d talking about topic z for a given q. After the document d^* is selected, PM-2 increases the "portion" of seats occupied by each of the topics z by its normalized relevance to d^*:

$$s_{z|q} \leftarrow s_{z|q} + \frac{P(d^*|z,q)}{\sum_{z'} P(d^*|z',q)}.$$

This process repeats until we get k documents for L_f or we are out of candidate documents. The order in which a document is appended to L_f determines its ranking.

We face two challenges in PM-2: it is non-trivial to get the aspect probability $v_{z|q}$ (i.e., $p(z|q)$), which is often set to be uniform, and it is non-trivial to compute $p(d|z,q)$, which usually requires explicit access to additional information. To address the first challenge, we compute $v_{z|q}$ by (7.2), such that (7.3) can be modified as:

$$qt[z|q] = \frac{p(z|q)}{2s_{z|q}+1} = \frac{\exp\{u_z + \frac{1}{2}\sigma_z^2\}}{(2s_{z|q}+1)\sum_{z'=1}^{T}\exp\{u_{z'} + \frac{1}{2}\sigma_{z'}^2\}}.$$

For the second challenge, instead of computing $P(d|z,q)$ explicitly, we modify $P(d|z,q)$ and apply Bayes' Theorem so that:

$$P(d|z,q) = \frac{p(z|d,q)p(d|q)}{p(z|q)} = \frac{p(z|d,q)p(d|q)}{v_{z|q}}. \tag{7.5}$$

Then we integrate the fused score generated by CombSUM into our model, i.e., we set:

$$p(d|q) \overset{rank}{=} f_{\text{CombSUM}}(d;q)$$

in (7.5). As a result, after applying (7.5) to (7.4), DDF selects a candidate document by:

$$\begin{aligned} d^* = \arg\max_{d\in\mathcal{R}} \ & \lambda \cdot qt[z^*|q] \cdot \frac{p(z^*|d,q) \cdot f_{\text{CombSUM}}(d;q)}{v_{z^*|q}} + \\ & (1-\lambda) \sum_{z\neq z^*} qt[z|q] \cdot \frac{p(z|d,q) \cdot f_{\text{CombSUM}}(d;q)}{v_{z|q}}, \end{aligned} \tag{7.6}$$

where $p(z|d;q)$ is the probability of document d belonging to topic z, which can easily be inferred in our DDF model by (8.10) (i.e., $p(z|d,q) = \theta_{d,z}$). Therefore, after applying (8.10) and (7.2), (7.6) can be rewritten as:

$$\begin{aligned} d^* = \arg\max_{d\in\mathcal{R}} \ & \lambda \cdot qt[z^*|q] \cdot \frac{\theta_{d,z^*} \cdot f_{\text{CombSUM}}(d;q)}{\exp\{\mu_z^* + \frac{1}{2}\sigma_z^{*2}\}} + \\ & (1-\lambda) \sum_{z\neq z^*} qt[z|q] \cdot \frac{\theta_{d,z} \cdot f_{\text{CombSUM}}(d;q)}{\exp\{\mu_z + \frac{1}{2}\sigma_z^2\}}, \end{aligned} \tag{7.7}$$

where it should be noted that we ignore the constant term:

$$\textstyle\sum_{z=1}^{T} \exp\{\mu_z + \frac{1}{2}\sigma_z^2\},$$

as it has no impact on selecting the candidate document d^*.

7.2 Experimental Setup

In this section, we describe our experimental setup; Section 7.2.1 lists our research questions; Section 7.2.2 lists the metrics and the baselines; Section 7.2.3 details the settings of the experiments.

7.2.1 Detailed research questions

We divide our main research question (RQ 4) into the following detailed research questions, and let these questions guiding the remainder of the chapter:

RQ 4.1 Do fusion methods help improve state-of-the-art search diversification methods? Do they help in terms of intent-aware precision, as our main metric? Does DDF beat standard and state-of-the-art fusion methods? (See Section 7.3.1 and Section 7.3.2 for answers.)

RQ 4.2 What is the effect on the diversification performance of DDF and fusion methods of the number of component lists? Does the contribution of fusion to diversification performance depend on the quality of the component lists? (See Section 7.3.3 for answers.)

RQ 4.3 Does DDF outperform the best diversification and fusion methods on each query? (See Section 7.3.4 for the answer.)

RQ 4.4 How do the rankings of DDF differ from those produced by other fusion methods? (See Section 7.3.5 for the answer.)

RQ 4.5 What is the effect on the diversification performance of DDF of the number of latent topics used by DDF? (See Section 7.3.6 for the answer.)

In order to answer our research questions we work with the runs submitted to the TREC 2009, 2010, 2011 and 2012 Web tracks, and the billion-page ClueWeb09 collection (see Section 3.2.3 for details).[1] There are two tasks in these tracks: an ad hoc search task and a search result diversification task (Clarke and Craswell, 2011; Clarke et al., 2009, 2010, 2012). We only focus on the diversification task, where the top-k documents returned should not only be relevant but also cover as many aspects as possible in response to a given query. In total, we have 200 ambiguous queries from the four years, with 2 queries (#95 and #100 in the 2010 edition) not having relevant documents. Typically, each query has 2 to 5 aspects, and some relevant documents are relevant to more than 2 aspects of the query.

Many of the runs submitted to these four years of the Web track for the diversification task were generated by state-of-the-art diversification methods. In total, we have 119, 88, 62 and 48 runs from the 2009, 2010, 2011 and 2012 editions, respectively.[2]

7.2.2 Evaluation metrics and baselines

We evaluate our component runs and fused runs using several standard metrics that are official evaluation metrics in the diversification tasks at TREC Web tracks (Clarke and Craswell, 2011; Clarke et al., 2009, 2010, 2012) and are widely used in the literature on search result diversification (Agrawal et al., 2009; Aktolga and Allan, 2013; Dang and Croft, 2012, 2013; Sakai et al., 2013; Santos et al., 2011): Prec-IA@k (Agrawal et al., 2009), MAP-IA@k (Agrawal et al., 2009), ERR-IA@k (Agrawal et al., 2009) and α-nDCG@k (Clarke et al., 2008b). The former two are set-based and indicate, respectively, the precision and mean average precision across all aspects of the query in the search results, whereas the remaining ones are cascade measures that penalize redundancy at each position in the ranked list based on how much of that information the user has already seen from documents at earlier ranks.

We follow published work on search result diversification and mainly compute the metric scores at depth 20. Statistical significance of observed differences between the performance of two runs is tested using a two-tailed paired t-test and is denoted using $^{\blacktriangle}$ (or $^{\blacktriangledown}$) for significant differences for $\alpha = .01$, or $^{\triangle}$ (and $^{\triangledown}$) for $\alpha = .05$.

[1] Available from `http://boston.lti.cs.cmu.edu/Data/clueweb09`.
[2] All runs are available from `http://trec.nist.gov`.

When assessing a fusion method X we will prefer fusion methods that are safe, where we say that X is *safe for metric* M if applying X to a set of component runs always yields a fused run that scores at least as high as the highest scoring component run in the set (according to M).

We consider several baselines. Two standard fusion methods (Lee, 1995), CombSUM and CombMNZ; two state-of-the-art fusion methods (Kozorovitsky and Kurland, 2011), ClustFuseCombSUM and ClustFuseCombMNZ; each year's best performing runs in the diversification tasks at the TREC Web track (Clarke and Craswell, 2011; Clarke et al., 2009, 2010, 2012), and state-of-the-art plain diversification methods, xQuAD (Santos et al., 2010a) and PM-2 (Dang and Croft, 2012). As DDF builds on both fusion and diversification methods, we also consider two fusion methods, CombSUMMMR and CombSUMPM-2, that integrate plain diversification methods MMR (Carbonell and Goldstein, 1998) and PM-2 into CombSUM for diversification, respectively.

7.2.3 Experiments

We report on five main experiments aimed at answering the research questions listed in Section 7.2.1. In our first experiment, aimed at determining whether fusion methods help diversification, we fuse the five top performing diversification result lists from the TREC Web 2009, 2010, 2011 and 2012 submitted runs (some lists are generated by the implementation of PM-2) by our baselines, viz., CombSUM, CombMNZ, ClustFuseCombSUM, ClustFuseCombMNZ, CombSUMMMR and CombSUMPM-2 (see Section 7.2.2). The performance of the baselines is compared against that of DDF.

Our second experiment is aimed at understanding the effect on the diversification performance of DDF and fusion methods of the number of component lists; we randomly sample $k \in \{2, 4, \ldots, 26\}$ component runs from the submitted runs in the TREC Web 2012 track and fuse them. We repeat the experiments 20 times and report the average results and the standard deviations. We also show one sample's result when fusing 4 runs.

Next, in order to understand how DDF outperforms the best component run and the fusion methods per query, our third experiment provides a query-level analysis. Our fourth experiment is aimed at understanding how the runs generated by DDF differ from those produced by other fusion methods; we zoom in on the differences between DDF and the next best performing fusion method, CombSUMPM-2, in terms of the documents (and aspects) retrieved by one, but not the other, or by both.

Finally, to understand the influence of the number of latent topics used in DDF, we vary the number of latent topics and assess the performance of DDF. We also use an oracle variant of DDF, called DDF2, where for every test query we consider as many latent topics as there are aspects according to the ground truth. The number of topics used in DDF is set to 10, unless stated otherwise.

7.3 Results

In Section 7.3.1 we examine the performance of baseline fusion methods on the diversification task, which we follow with a section on the performance of DDF in Section 7.3.2.

Section 7.3.3 details the effect of the number of lists; Section 7.3.4 provides a query-level analysis; Section 7.3.5 zooms in on the effect on ranking of DDF compared to the next best fusion method; Section 7.3.6 examines the effect of the number of latent topics on DDF.

7.3.1 Performance of baseline fusion methods

In Table 7.2 we list the diversity scores of the baseline fusion methods on the diversity task: CombSUM, CombMNZ, ClustFuseCombSUM, ClustFuseCombMNZ, CombSUMMMR, CombSUMPM-2, with the 5 best performing component lists from the TREC Web 2009, 2010, 2011 and 2012 tracks, respectively.[3] For all metrics and in all years, almost all baseline fusion methods outperform the state-of-the-art diversification methods, and in many cases significantly so. Note, however, that none of the baseline methods is safe in the sense defined in Section 7.2.2. Additionally, Table 7.3 shows the diversity scores of the baseline fusion methods when we fuse 4 randomly sampled runs from the 2012 data set, which confirms that fusion does help diversification.

7.3.2 The performance of DDF

Inspired by the success of baseline fusion methods on the diversification task, we now consider our newly proposed fusion method, DDF. Returning to Tables 7.2 and 7.3, two types of conclusion emerge. First, DDF outperforms all component runs (note that component runs in Table 7.2 are the best runs in the tracks), on all metrics, for all years. In other words, it is *safe* in the sense defined in Section 7.2.2. The difference between DDF and the best performing component run is always significant. We believe that the strong performance of DDF is due to the fact that DDF not only focuses on improving the relevance score of fused run but also explicitly tries to diversify the fused run.

Second, DDF outperforms all baseline fusion methods, on all metrics. In many cases, CombSUMPM-2 and CombSUM yield the second and third best performance, respectively, but DDF outperforms them in every case, and often significantly so. DDF can beat CombSUMPM-2 as it tackles two main challenges in PM-2 (see part III in Section 7.1), although they build on the same framework. CombSUMMMR follows a similar strategy as DDF but its performance is worse than that of DDF. This is due to the fact that MMR models documents as if they are centered around a single topic only. It is clear from Tables 7.2 and 7.3 that cluster-based data fusion methods (ClustFuseCombSUM, ClustFuseCombMNZ) sometimes perform a little worse than the standard fusion method they build on (CombSUM, CombMNZ). This is because cluster-based fusion focuses on relevance of the documents rather than on diversification.

[3]The run "PM-2 (TREC)" is the run that utilizes aspect information from the ground truth in the PM-2 model and the run "PM-2 (engine)" is produced using information from a commercial search engine. The run "xQuAD (uogTrX)" is a uogTrX TREC edition run generated using the xQuAD algorithm; see (Limsopatham et al., 2012).

Table 7.2: Performance obtained using the 2009–2012 editions of the TREC Web tracks. The best performing run per metric per year is in boldface. Statistically significant differences between the fusion method and the best component run, between DDF and CombSUM, and between DDF and CombSUMPM-2, are marked in the upper right hand corner of the score, in the upper left hand corner of DDF's score, and in the lower left hand corner of DDF's score, respectively.

		Prec-IA	MAP-IA	α-nDCG	ERR-IA
2012	DFalah120A	.3241	.0990	.5291	.4259
	DFalah120D	.3241	.0990	.5291	.4259
	xQuAD (uogTrA44xi)	.3349	.1345	.5917	.4873
	xQuAD (uogTrA44xu)	.3504	.1360	.6061	.5048
	xQuAD (uogTrB44xu)	.3389	.1339	.5795	.4785
	ClustFuseCombMNZ	.3533	.1488$^{\blacktriangle}$	.6010	.5105
	ClustFuseCombSUM	.3545	.1495$^{\blacktriangle}$	.5965	.5049
	CombSUMMMR	.3558	.1544$^{\blacktriangle}$	.6106	.5115
	CombSUMPM-2	.3718$^{\blacktriangle}$	.1826$^{\blacktriangle}$	.6228$^{\blacktriangle}$	.5179$^{\triangle}$
	CombMNZ	.3663$^{\blacktriangle}$	.1785$^{\blacktriangle}$	.6154$^{\triangle}$	.5153$^{\triangle}$
	CombSUM	.3592$^{\triangle}$	.1767$^{\blacktriangle}$	.6114$^{\triangle}$	.5126$^{\triangle}$
	DDF	**.3904**$^{\blacktriangle\blacktriangle}$	**.1910**$^{\blacktriangle\blacktriangle}$	**.6334**$^{\blacktriangle\blacktriangle}$	**.5266**$^{\blacktriangle\blacktriangle}$
2011	ICTNET11ADR2	.2993	.1328	.5725	.4658
	umassGQdist	.3003	.1313	.5513	.4530
	xQuAD (uogTrA45Nmx2)	.3039	.1365	.6298	.5284
	xQuAD (uogTrA45Vmx)	.3030	.1323	.6304	.5238
	UWatMDSdm	.3214	.1350	.5979	.4875
	ClustFuseCombMNZ	.3303$^{\blacktriangle}$	.1757$^{\blacktriangle}$	.6221$^{\triangledown}$	.5001
	ClustFuseCombSUM	.3296$^{\triangle}$	.1775$^{\blacktriangle}$	.6307	.5110
	CombSUMMMR	.3395$^{\blacktriangle}$	.1830$^{\blacktriangle}$	.6341	.5107
	CombSUMPM-2	.3450$^{\blacktriangle}$	.2024$^{\blacktriangle}$	.6448$^{\blacktriangle}$	.5196
	CombMNZ	.3413$^{\blacktriangle}$	.1943$^{\blacktriangle}$	.6430$^{\blacktriangle}$	.5209
	CombSUM	.3376$^{\blacktriangle}$	.1966$^{\blacktriangle}$	.6423$^{\blacktriangle}$	.5216
	DDF	**.3596**$^{\blacktriangle\blacktriangle}$	**.2102**$^{\blacktriangle\blacktriangle}$	**.6496**$^{\blacktriangle}$	**.5295**$^{\blacktriangle}$
2010	CSE.pm2.run	.1832	.0351	.4165	.3052
	cmuWi10D	.1879	.0599	.3452	.2484
	xQuAD (uogTrA42x)	.1845	.0529	.3558	.2454
	PM-2 (engine)	.2009	.0414	.3660	.2581
	PM-2 (TREC)	.2026	.0430	.4449	.3320
	ClustFuseCombMNZ	.2105	.0845$^{\blacktriangle}$	.4313	.3221
	ClustFuseCombSUM	.2072	.0825$^{\blacktriangle}$	.4257$^{\triangledown}$	.3148$^{\triangledown}$
	CombSUMMMR	.2115$^{\triangle}$	.0836$^{\blacktriangle}$	.4366	.3189
	CombSUMPM-2	.2129$^{\blacktriangle}$	.0839$^{\blacktriangle}$	.4379	.3193
	CombMNZ	.2177$^{\blacktriangle}$	.0899$^{\blacktriangle}$	.4471	.3411$^{\triangle}$
	CombSUM	.2159	.0875$^{\blacktriangle}$	.4454	.3350
	DDF	**.2285**$^{\blacktriangle\blacktriangle}$	**.0910**$^{\triangle\blacktriangle}$	**.4627**$^{\blacktriangle\blacktriangle}$	**.3406**$^{\blacktriangle\blacktriangle}$
2009	NeuDiv1	.1343	.0458	.2781	.1705
	NeuDivW75	.1239	.0397	.2501	.1598
	xQuAD(uogTrDPCQcdB)	.1302	.0463	.2968	.1848
	xQuAD (uogTrDYCcsB)	.1268	.0444	.3081	.1922
	uwgym	.1224	.0456	.2798	.1701
	ClustFuseCombMNZ	.1381	.0681$^{\blacktriangle}$	.3076	.1937
	ClustFuseCombSUM	.1379	.0680$^{\blacktriangle}$	.3223$^{\blacktriangle}$	.2005
	CombSUMMMR	.1424$^{\triangle}$	.0682$^{\blacktriangle}$	.3343$^{\blacktriangle}$	.2028$^{\triangle}$
	CombSUMPM-2	.1588$^{\blacktriangle}$	.0754$^{\blacktriangle}$	.3887$^{\blacktriangle}$	.2674$^{\blacktriangle}$
	CombMNZ	.1400$^{\triangle}$	.0666$^{\blacktriangle}$	.3343$^{\blacktriangle}$	.2033$^{\triangle}$
	CombSUM	.1400$^{\triangle}$	.0664$^{\blacktriangle}$	.3482$^{\blacktriangle}$	.2080$^{\triangle}$
	DDF	**.1631**$^{\blacktriangle\blacktriangle}$	**.0731**$^{\blacktriangle\blacktriangle}$	**.4005**$^{\blacktriangle\blacktriangle}$	**.2713**$^{\blacktriangle}$

Table 7.3: Performance obtained using the 2012 editions of the TREC Web track. The best performing run per metric is in boldface. Other notational conventions as in Table 7.2.

		Prec-IA	MAP-IA	α-nDCG	ERR-IA
2012	QUTparaBline	.2261	.0639	.5270	.4185
	xQuAD (uogTrA44xl)	.2957	.1077	.5161	.4009
	utw2012c1	.1637	.0439	.5075	.4046
	PM-2 (TREC)	.2631	.0601	.5245	.4155
	ClustFuseCombMNZ	.2735$^{\triangledown}$	.1155$^{\blacktriangle}$	.5717$^{\blacktriangle}$	.4608$^{\blacktriangle}$
	ClustFuseCombSUM	.2752	.1172$^{\blacktriangle}$	.5726$^{\blacktriangle}$	.4674$^{\blacktriangle}$
	CombSUMMMR	.2783$^{\triangledown}$	.1189$^{\blacktriangle}$	.5799$^{\blacktriangle}$	.4633$^{\blacktriangle}$
	CombSUMPM-2	.2934	.1305$^{\blacktriangle}$	.6013$^{\blacktriangle}$	.4877$^{\blacktriangle}$
	CombMNZ	.2864	.1267$^{\blacktriangle}$	.5851$^{\blacktriangle}$	.4708$^{\blacktriangle}$
	CombSUM	.2884	.1275$^{\blacktriangle}$	.5944$^{\blacktriangle}$	.4803$^{\blacktriangle}$
	DDF	**$^{\blacktriangle}_{\blacktriangle}$.3193$^{\blacktriangle}$**	**$^{\blacktriangle}_{\blacktriangle}$.1409$^{\blacktriangle}$**	**$^{\blacktriangle}_{\triangle}$.6107$^{\blacktriangle}$**	**$^{\blacktriangle}_{\triangle}$.4919$^{\blacktriangle}$**

7.3.3 Effect of the number of component lists

Next, we zoom in on DDF. In particular, we explore the effect of varying the number of lists to be fused on its performance. Fig. 7.2 shows the fusion results of randomly sampling $k \in \{2, 4, \ldots, 26\}$ lists from the 48 runs submitted to the TREC Web 2012 track plus the PM-2 runs (due to space limitations, we only report results using the 2012 runs; the findings on other years are qualitatively similar). For each k, we repeat the experiment 20 times and report on the average scores and the corresponding standard deviations indicated by the error bars in the figure. We use CombSUM as a representative example for comparison with DDF, as the results of other baseline fusion methods are worse or have qualitatively similar results to those of CombSUM.

As shown in Fig. 7.2, DDF always outperforms CombSUM in terms of the Prec-IA, α-nDCG and ERR-IA evaluation metrics and the performance gaps remain almost unchanged, in absolute terms, no matter how many component lists are fused. One reason for this is that as DDF builds on CombSUM, it inherits the merits of the fusion method, and more importantly, at the same time it tries to infer latent topics and rerank the high ranked documents in terms of novelty of the documents. For the MAP-IA metric, however, the gaps increase with more component lists being fused. The performance of both DDF and CombSUM increases faster when the number of component lists increases but is ≤ 10 than when the number of component lists is > 10, for all the metrics. This seems to be inherent to the underlying CombSUM method and is due to the fact that with smaller numbers of component lists, there is simply more space available at depth 20 to obtain improvements than with larger numbers of component lists.

7.3.4 Query-level analysis

We take a closer look at per test query improvements of DDF over the best baseline fusion run when fusing the best 5 runs in 2012, viz., CombSUMPM-2, which outperforms the best component list. Fig. 7.3 shows the per query performance differences in terms of

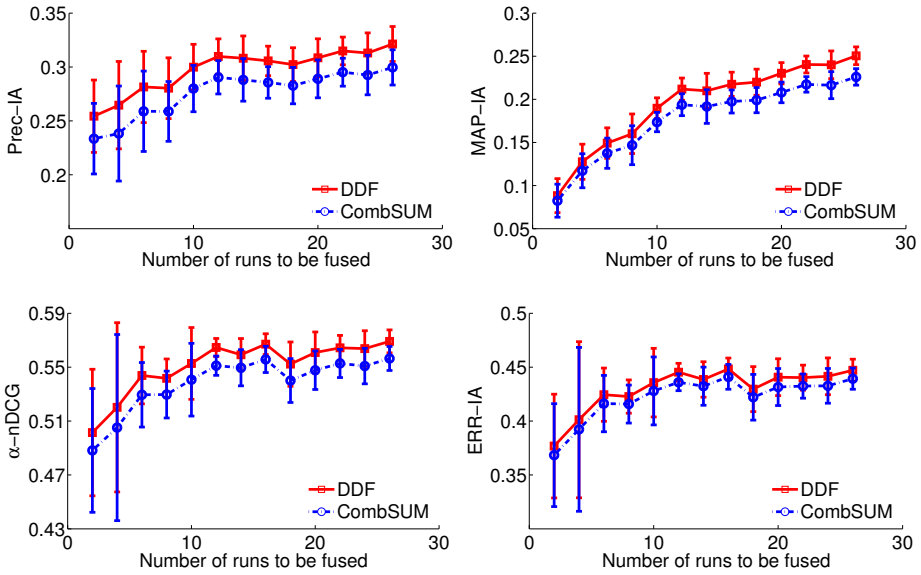


Figure 7.2: Effect on performance (in terms of Prec-IA, MAP-IA, α-nDCG and ERR-IA) of the number of component lists, using runs sampled from the TREC 2012 Web track. We plot averages and standard deviations. Note: the figures are not to the same scale.

Prec-IA, MAP-IA, α-nDCG and ERR-IA, respectively, of DDF against CombSUMPM-2. DDF achieves performance improvements for many queries when compared against CombSUMPM-2, although the differences are sometimes relatively small.

In a very small number of cases, DDF performs poorer than CombSUMPM-2. This appears to be due to the fact that DDF "over-diversifies" documents in runs produced by CombSUM that have very few relevant document to start with, so that DDF ends up promoting different but non-relevant documents.

7.3.5 Zooming in on Prec-IA@k

Next, we zoom in on one of the metrics that shows the biggest relative differences between DDF and the next best performing fusion method, Prec-IA, so as to understand how the runs generated by DDF differ from those by other fusion-based methods. Here, again, we use CombSUMPM-2 as a representative, as it tends to outperform or equal the other fusion methods. Specifically, we report changes in the number of relevant documents for DDF against CombSUMPM-2 when fusing the 2012 runs in Table 7.2 in 2012; see Fig. 7.4. Red bars indicate the number of relevant documents that appear in the run of DDF but not the run of CombSUMPM-2, white bars indicate the number of relevant documents in both runs, whereas blue bars indicate the number of relevant documents that appear not in DDF but in CombSUMPM-2; topics are ordered first by the size of the red bar, then the size of the white bar, and finally the size of the blue bar.

Clearly, the differences between DDF and CombSUMPM-2 in the top 5 and 10 are more limited than the differences in the top-15 and 20, but in all cases DDF outperforms

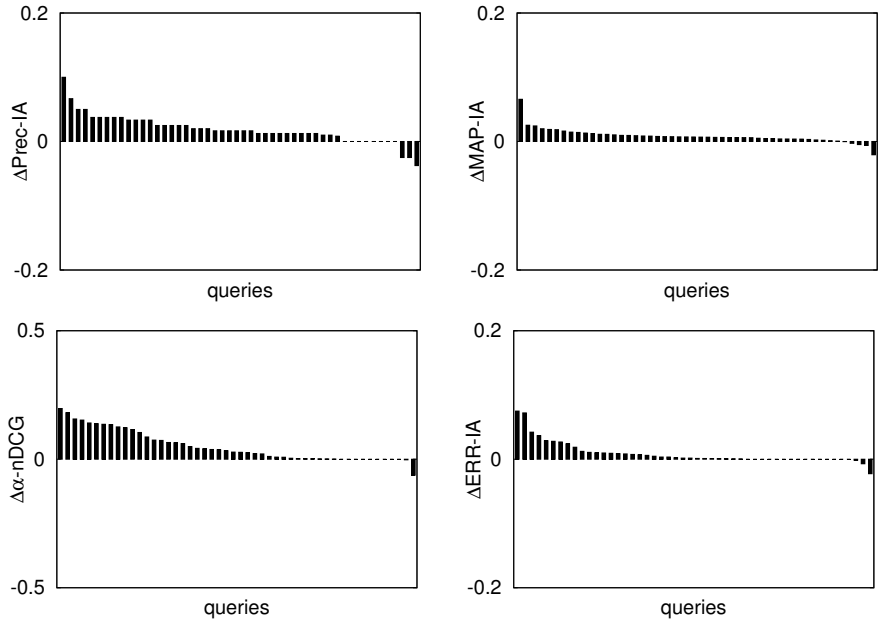


Figure 7.3: Per query performance differences of DDF against CombSUMPM-2 (second row). The figures shown are for fusing the runs in TREC Web 2012 track, for Prec-IA@20, MAP-IA@20, α-nDCG@20 and ERR-IA@20 (from left to right). A bar extending above the center of a plot indicates that DDF outperforms CombSUMPM-2, and vice versa for bars below the center.

CombSUMPM-2. E.g., in total there are 45 more relevant documents in the top 20 of the run produced by DDF than those in the CombSUMPM-2 run (49 relevant documents in DDF but not in CombSUMPM-2, 4 relevant documents in CombSUMPM-2 but not in DDF). We examine the matter further by comparing the Prec-AI@5, 10, 15, 20 scores of the DDF and CombSUMPM-2 runs for the 2012 data; see Table 7.4. The differences at small depths (5, 10) are weakly statistically significant while those at bigger depths are significant, confirming our observations in Fig. 7.4; we also find that DDF statistically significantly outperforms CombSUMPM-2 in terms of Prec-IA scores at depth 5, 10, 15 and 20, which again confirms the above observations based on Fig. 7.4.

Table 7.4: Prec-IA@5, 10, 15, 20 performance comparison between CombSUMPM-2 and DDF. A statistically significant difference between DDF and CombSUMPM-2 is marked in the upper left hand corner of the DDF score.

Prec-IA@	5	10	15	20
CombSUMPM-2	.4367	.4066	.3887	.3718
DDF	$^{\triangle}$.4555	$^{\triangle}$.4194	$^{\blacktriangle}$.4060	$^{\blacktriangle}$.3904

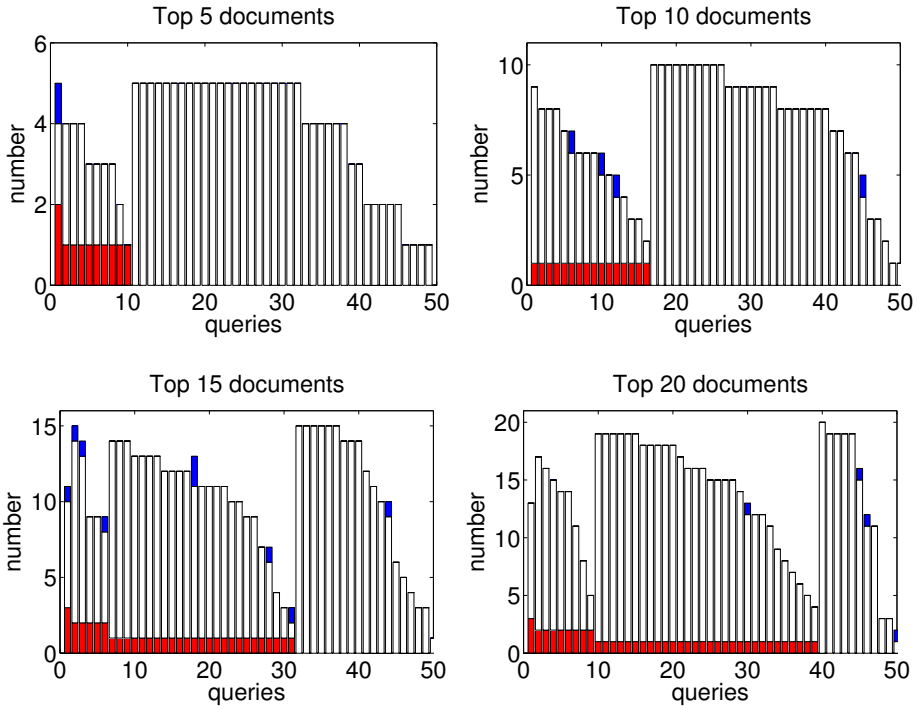


Figure 7.4: How runs produced by DDF and CombSUMPM-2 differ. Red, white, blue bars indicate the number of relevant documents that appear in DDF but not in CombSUMPM-2, in both runs and not in DDF but in CombSUMPM-2, respectively, at corresponding depth k (for $k = 5, 10, 15, 20$). Figures should be viewed in color.

7.3.6 Effect of the number of topics

Finally, we examine the effect on the overall performance of the number of latent topics used in DDF, and contrast the performance of DDF with varying number of latent topics against DDF2, CombSUM and CombSUMPM-2. Here, DDF2 is the same algorithm as DDF except that for every test query it considers as many latent topics as there are aspects according to the ground truth. We use DDF2, DDF, CombSUM and CombSUMPM-2 to fuse the component result runs listed in Table 7.2 in 2012 as an example. We vary the number of latent topics in DDF from 2 to 16. See Fig. 7.5.

When the number of latent topics used in DDF increases from 2 to 6, the performance of DDF increases dramatically. When only 2 latent topics are used, the performance is worse than that of CombSUM and CombSUMPM-2; e.g., Prec-IA@20 for DDF is 0.3404, while the scores of CombSUM and CombSUMPM-2 are 0.3592 and 0.3718, respectively. In contrast, when the number of latent topics varies between 8 to 16, the performance of DDF seems to level off. This demonstrates another merit of our fusion model, DDF: it is robust and not sensitive to the number of latent topics once the number of latent topics is "large enough." Another important finding from Fig. 7.5 is that DDF2 always enhances the performance of DDF, CombSUM and CombSUMPM-2, for

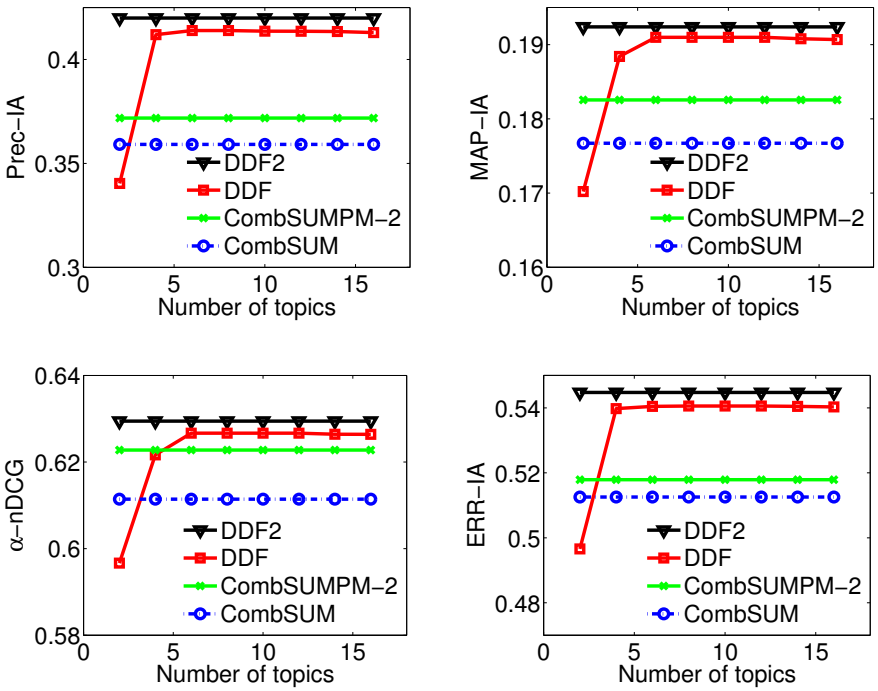


Figure 7.5: Performance comparison between DDF2, DDF, CombSUMPM-2 and CombSUM when varying the number of latent topics used in DDF. Note: the figures are not to be the same scale.

all metrics, which demonstrates the fact that latent topics can enhance the performance. The performance differences between DDF2 and DDF are quite marginal and not statistically significant. We leave it as future work to dynamically estimate the number of aspects (and latent topics) of an incoming query and to use this estimate in DDF.

7.4 Conclusion

Most previous work on search result diversification focuses on the content of the documents returned by an ad hoc algorithm to diversify the results implicitly or explicitly, i.e., using implicit or explicit representations of aspects. In this chapter we have adopted a different perspective on the search result diversification problem, based on data fusion.

We proposed to use traditional unsupervised and state-of-the-art data fusion methods, CombSUM, CombMNZ, ClustFuseCombSUM, ClustFuseCombMNZ, CombSUMMMR and CombSUMPM-2 to diversify result lists. This led to the insight that fusion does aid diversification. We also proposed a fusion-based diversification method, DDF, which infers latent topics from ranked lists of documents produced by a standard fusion method, and combines this with a state-of-the-art result diversification model. Our experiments provide answers to the main research question raised at the beginning of this

chapter:

RQ 4 Can data fusion help search result diversification?

To answer the main research question, we worked with the runs submitted to the TREC 2009, 2010 and 2012 Web tracks, and the billion-page ClueWeb09 official collection in the TREC. In the experiments, we considered a number of baselines, including plain data fusion methods, e.g., CombSUM, plain search result diversification methods, e.g., PM-2, and the mixed methods, e.g., CombSUMPM-2. We found that data fusion approaches outperform state-of-the-art search result diversification algorithms, with DDF invariably giving rise to the highest scores on all of the metrics that we have considered in this chapter. DDF was shown to behave well with different numbers of component lists. We also found that DDF is insensitive to the number of latent topics of a query, once a sufficiently large number was chosen, e.g., 10.

As to future work, we aim to incorporate into DDF methods for automatically estimating the number of aspects, which will be used to set the number of latent topics. We also plan to use data fusion methods BurstFuseX and TimeRA proposed in Chapters 4 and 5 for diversification and compare them to DDF. The last and third part of DDF is based on a particular choice of method, viz. PM-2, and we only apply rank-based fusion methods for diversification. In future work we plan to compare these choices with alternative choices, and apply other fusion alternatives, e.g., score-based fusion methods.

In Chapters 4 and 5 we focused on data fusion methods for microblog search, and in Chapter 6 we focused on data fusion methods for ad hoc search. In this chapter, we considered alternative application of data fusion in search result diversification, where only document are token into account. In the next chapter, we will continue the research of search result diversification and investigate how to utilize users' personalized information for search result diversification.

7.A Gibbs Sampling Derivation for DDF Model

We begin with the joint distribution $P(\mathbf{w}, \mathbf{f}, \mathbf{z}|\alpha, \beta, \mu, \sigma, \mathbf{L})$ and use conjugate priors to simplify the integrals. Notation defined in Section 7.1.

$$
\begin{aligned}
&P(\mathbf{w}, \mathbf{f}, \mathbf{z}|\alpha, \beta, \mu, \sigma, \mathbf{L}, q) = P(\mathbf{w}|\mathbf{z}, \beta)p(\mathbf{f}|\mu, \sigma, \mathbf{z}, \mathbf{L})P(\mathbf{z}|\alpha) \\
&= \int P(\mathbf{w}|\mathbf{\Phi}, \mathbf{z})p(\mathbf{\Phi}|\beta)d\mathbf{\Phi} \times p(\mathbf{f}|\mu, \sigma, \mathbf{z}, \mathbf{L}, q) \int P(\mathbf{z}|\mathbf{\Theta})P(\mathbf{\Theta}|\alpha)d\mathbf{\Theta} \\
&= \int \prod_{d=1}^{|\mathcal{C}_\mathbf{L}|} \prod_{i=1}^{N_d} P(w_{di}|\phi_{z_{di}}) \prod_{z=1}^{T} p(\phi_z|\beta)d\mathbf{\Phi} \\
&\quad \times \prod_{d=1}^{|\mathcal{C}_\mathbf{L}|} \prod_{i=1}^{N_d} p(f_{di}|\mu_{z_{di}}, \sigma_{z_{di}}, \mathbf{L}, q) \\
&\quad \times \int \prod_{d=1}^{|\mathcal{C}_\mathbf{L}|} \left(\prod_{i=1}^{N_d} P(z_{di}|\theta_d)p(\theta_d|\alpha) \right) d\mathbf{\Theta} \\
&= \int \prod_{z=1}^{T} \prod_{v=1}^{V} \phi_{z_v}^{n_{z_v}} \prod_{z=1}^{T} \left(\frac{\Gamma(\sum_{v=1}^{V} \beta_v)}{\prod_{v=1}^{V} \Gamma(\beta_v)} \prod_{v=1}^{V} \phi_{z_v}^{\beta_v - 1} \right) d\mathbf{\Phi} \\
&\quad \times \prod_{d=1}^{|\mathcal{C}_\mathbf{L}|} \prod_{i=1}^{N_d} p(f_{di}|\mu_{z_{di}}, \sigma_{z_{di}}, \mathbf{L}, q) \\
&\quad \times \int \prod_{d=1}^{|\mathcal{C}_\mathbf{L}|} \prod_{z=1}^{T} \theta_{d_z}^{m_{d_z}} \prod_{d=1}^{|\mathcal{C}_\mathbf{L}|} \left(\frac{\Gamma(\sum_{z=1}^{T} \alpha_z)}{\prod_{z=1}^{T} \Gamma(\alpha_z)} \prod_{z=1}^{T} \theta_{d_z}^{\alpha_z - 1} \right) d\mathbf{\Theta} \\
&= \left(\frac{\Gamma(\sum_{v=1}^{V} \beta_v)}{\prod_{v=1}^{V} \Gamma(\beta_v)} \right)^T \left(\frac{\Gamma(\sum_{z=1}^{T} \alpha_z)}{\prod_{z=1}^{T} \Gamma(\alpha_z)} \right)^{|\mathcal{C}_\mathbf{L}|} \\
&\quad \times \prod_{d=1}^{|\mathcal{C}_\mathbf{L}|} \prod_{i=1}^{N_d} p(f_{di}|\mu_{z_{di}}, \sigma_{z_{di}}, \mathbf{L}, q) \\
&\quad \times \prod_{z=1}^{T} \frac{\prod_{v=1}^{V} \Gamma(n_{z_v} + \beta_v)}{\Gamma(\sum_{v=1}^{V}(n_{z_v} + \beta_v))} \prod_{d=1}^{|\mathcal{C}_\mathbf{L}|} \frac{\prod_{z=1}^{T} \Gamma(m_{d_z} + \alpha_z)}{\Gamma(\sum_{z=1}^{T}(m_{z_d} + \alpha_z))}
\end{aligned}
$$

Using the chain rule, we can obtain the conditional probability conveniently,

$$
\begin{aligned}
&P(z_{di}|\mathbf{w}, \mathbf{f}, \mathbf{z}_{-di}, \alpha, \beta, \mu, \sigma, \mathbf{L}, q) \\
&= \frac{P(z_{di}, w_{di}, f_{di}|\mathbf{w}_{-di}, \mathbf{f}_{-di}, \mathbf{z}_{-di}, \alpha, \beta, \mu, \sigma, \mathbf{L}, q)}{P(w_{di}, f_{di}|\mathbf{w}_{-di}, \mathbf{f}_{-di}, \mathbf{z}_{-di}, \alpha, \beta, \mu, \sigma, \mathbf{L}, q)} \\
&= \frac{P(\mathbf{w}, \mathbf{f}, \mathbf{z}|\alpha, \beta, \mu, \sigma, \mathbf{L}, q)}{P(\mathbf{w}, \mathbf{f}, \mathbf{z}_{-di}|\alpha, \beta, \mu, \sigma, \mathbf{L}, q)} \\
&\text{because } z_{di} \text{ depends only on } w_{di} \text{ and } f_{di} \\
&\propto \frac{P(\mathbf{w}, \mathbf{f}, \mathbf{z}|\alpha, \beta, \mu, \sigma, \mathbf{L}, q)}{P(\mathbf{w}_{-di}, \mathbf{f}_{-di}, \mathbf{z}_{-di}, |\alpha, \beta, \mu, \sigma, \mathbf{L}, q)} \\
&\propto (m_{dz_{di}} + \alpha_{z_{di}} - 1) \frac{n_{z_{di} w_{di}} + \beta_{w_{di}} - 1}{\sum_{v=1}^{V}(n_{z_{di} v} + \beta_v) - 1}
\end{aligned}
$$

$$
\begin{aligned}
& \times \frac{1}{f_{di}\sigma_{z_{di}}\sqrt{2\pi}} \exp\{-\frac{(\ln f_{di} - \mu_{z_{di}})^2}{2\sigma_{z_{di}}^2}\} \\
\propto \quad & (m_{dz_{di}} + \alpha_{z_{di}} - 1)\frac{n_{z_{di}w_{di}} + \beta_{w_{di}} - 1}{\sum_{v=1}^{V}(n_{z_{di}v} + \beta_v) - 1} \\
& \times \frac{1}{f_{\mathrm{X}}(d|\mathbf{L}, q)\sigma_{z_{di}}\sqrt{2\pi}} \exp\{-\frac{(\ln f_{\mathrm{X}}(d|\mathbf{L}, q) - \mu_{z_{di}})^2}{2\sigma_{z_{di}}^2}\},
\end{aligned}
$$

where $f_{\mathrm{X}}(d|\mathbf{L}, q) \in (0, +\infty)$ is a fusion score generated by a standard fusion method f_{X} for document $d \in \mathcal{C}_{\mathbf{L}}$ given the observation of lists $\mathbf{L}$ to be merged and query q. We use $f_{\mathrm{CombSUM}}(d|\mathbf{L}, q)$.

Since the data fusion score of a token that appears in d when fusing all the lists in $\mathbf{L}$ given a query q and the latent topics of which is z_{di}, is drawn from log-normal distributions, sparsity is not a big problem for parameter estimation of both $\mu_{z_{di}}$ and $\sigma_{z_{di}}$. For simplicity, we update both $\mu_{z_{di}}$ and $\sigma_{z_{di}}$ after each Gibbs sample iteration by maximum likelihood estimation:

$$
\begin{aligned}
\hat{\mu}_{z_{di}} &= \frac{\sum_{d'=1}^{|\mathcal{C}_{\mathbf{L}}|}\sum_{i' \wedge (z_{d'i'}=z_{di})}^{N_d} \ln f_{d'i'}}{n_{z_{di}}} \\
&= \frac{\sum_{d'=1}^{|\mathcal{C}_{\mathbf{L}}|}\sum_{i' \wedge (z_{d'i'}=z_{di})}^{N_d} \ln f_{\mathrm{X}}(d'|\mathbf{L}, q)}{n_{z_{di}}} \\
\hat{\sigma}_{z_{di}}^2 &= \frac{\sum_{d'=1}^{|\mathcal{C}_{\mathbf{L}}|}\sum_{i' \wedge (z_{d'i'}=z_{di})}^{N_d} (\ln f_{d'i'} - \hat{\mu})^2}{n_{z_{di}}} \\
&= \frac{\sum_{d'=1}^{|\mathcal{C}_{\mathbf{L}}|}\sum_{i' \wedge (z_{d'i'}=z_{di})}^{N_d} (\ln f_{\mathrm{X}}(d'|\mathbf{L}, q) - \hat{\mu})^2}{n_{z_{di}}}
\end{aligned}
$$

8

Personalized Diversification

In the previous chapter, we have explored how to utilize data fusion for search result diversification where we only focus on the content of the documents and no user's information is taken into account. In this chapter, we continue the research question of search result diversification, but also integrate user's information for diversification. In both search result diversification and personalized web search, an issued query is often viewed as an incomplete expression of a user's underlying need (Shen et al., 2005). Unlike search result diversification, where the system accepts and adapts its behavior to a situation of uncertainty, personalized web search strives to change this situation by enhancing the system's knowledge about users' information needs. Rather than aiming to satisfy as many users as possible, personalization aims to build a sense of who the user is, and maximizes the satisfaction of a specific user (Vallet and Castells, 2012).

Although different, diversification and personalization are not incompatible and do not have mutually exclusive goals (Shi et al., 2012). Search results generated by diversification techniques should be more diverse when a user's preferences are unrelated to the query. Likewise, personalization can improve the effectiveness of aspect weighting in diversification, by favoring query interpretations that are predicted to be more related to each specific user (Vallet and Castells, 2012).

In this chapter we study the problem of *personalized diversification of search results*, with the goal of enhancing both diversification and personalization performances. The problem has previously been investigated by Radlinski and Dumais (2006) and Vallet and Castells (2012). They have presented a number of effective *unsupervised learning* approaches that combine both personalization and diversification components to tackle the problem. To further improve the performance, we propose a *supervised learning* approach.

There are a couple of advantages to considering a supervised learning approach. Such approaches can leverage useful information underlying labeled training data, apply existing optimization techniques to the problem and are easier to adaptation. Of course, they also have disadvantages, one of which is that it is expensive to create training data for supervised learning methods. This is, however, a shortcoming for any supervised learning strategy and we leave it as future work. Accordingly, we are interested in answering:

RQ 5 How to enhance both diversification and personalization performance at the same time in a supervised way?

To answer the question, we formulate the task of personalized search result diversification as a problem of predicting a diverse set of documents given a specific user and a query. Specifically, we formulate a discriminant based on maximizing search result diversification, and perform training using the well-known structured support vector machines (SSVMs) framework (Tsochantaridis et al., 2005). The main idea is first to propose a user-interest LDA-style (Blei et al., 2003, Latent Dirichlet Allocation) topic model, from which we can infer a per-document multinomial distribution over topics and determine whether a document can cater for a specific user. Then, during training we use features extracted directly from the tokens' statistical information in the documents and those utilized by unsupervised personalized diversification algorithms, and, more importantly, those generated from our proposed topic model. Additionally, two types of constraints in SSVMs are explicitly defined to enforce the search results to be both diverse and relevant to a user's personal interest.

We evaluate our proposed approach on a publicly available personalized diversification dataset and compare it to unsupervised approaches, that focus on either personalization or diversification alone, to combined approaches like those in (Radlinski and Dumais, 2006) and (Vallet and Castells, 2012), and to two standard structured learning approaches (Yue and Joachims, 2008; Yue et al., 2007). The three main contributions of our work in this chapter are as follows:

i. We tackle the problem of personalized diversification of search results differently, using a supervised learning method.
ii. We propose a user-interest latent topic model to capture a user's interest and infer per-document multinomial distributions over topics.
iii. We explicitly enforce diversity and personalization through two types of constraints in structured learning for personalized diversification.

The remainder of the chapter is organized as follows. We frame the challenge as a structured learning problem in Section 8.1. Section 8.2 describes our structured learning for personalized diversification. Section 8.4 describes experimental setup. Section 8.5 is devoted to our results and we conclude in Section 8.6.

8.1 The Learning Problem

Let $\mathbf{u} = \{d_1, \ldots, d_{|\mathbf{u}|}\} \in \mathcal{U}$ be a set of documents of size $|\mathbf{u}|$, which a user u is interested in. For each query q, we assume that we are given $\mathbf{u}$ and a set of candidate documents $\mathbf{x} = \{x_1, \ldots, x_{|\mathbf{x}|}\} \in \mathcal{X}$, where $\mathcal{X}$ denotes the set of all possible document sets. Our task is to select a subset $\mathbf{y} \in \mathcal{Y}$ of K documents from $\mathbf{x}$ that maximizes the performance of personalized search result diversification given q and $\mathbf{u}$, where we let $\mathcal{Y}$ denote the space of predicted subsets $\mathbf{y}$. Following the standard machine learning setup, we formulate our task as learning a hypothesis function $h : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{Y}$ to predict a $\mathbf{y}$ given $\mathbf{x}$ and $\mathbf{u}$. To this end, we assume that a set of labeled training data is available:

$$\{(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)}) \in \mathcal{X} \times \mathcal{U} \times \mathcal{Y} : i = 1, \ldots, N\},$$

where $\mathbf{y}^{(i)}$ is the ground-truth subset of K documents from $\mathbf{x}^{(i)}$, and $\mathbf{u}^{(i)}$ is the set of documents that user u_i is interested in, and N is the size of the training data. We aim to

find a function h such that the empirical risk $R_S^{\Delta}(h) = \frac{1}{N}\sum_{i=1}^{N}\Delta(\mathbf{y}^{(i)}, h(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}))$ can be minimized, where we quantify the quality of a prediction by considering a loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^{+}$ that measures the penalty of choosing $\bar{\mathbf{y}} = h(\mathbf{x}^{(i)}, \mathbf{u}^{(i)})$. Here, given the ground truth $\mathbf{y}$, viz., the ground truth ranking of relevant documents, and the prediction $\bar{\mathbf{y}}$, viz., the ranking of predicted documents, we define the loss function based on a diversity metric, α-nDCG (Clarke et al., 2008b) (other diversity metrics are possible but we obtain the best performance when adopting this metric), as:

$$\Delta(\mathbf{y}, \bar{\mathbf{y}}) = 1 - \alpha\text{-nDCG}(\mathbf{y}, \bar{\mathbf{y}}). \tag{8.1}$$

We focus on hypothesis functions that are parameterized by a weight vector $\mathbf{w}$, and thus wish to find $\mathbf{w}$ to minimize the risk, $R_S^{\Delta}(\mathbf{w}) \equiv R_S^{\Delta}(h(\cdot; \mathbf{w}))$. We let a discriminant $\mathcal{F} : \mathcal{X} \times \mathcal{U} \times \mathcal{Y} \rightarrow \mathbb{R}^{+}$ compute how well the prediction $\bar{\mathbf{y}}$ fits for $\mathbf{x}$ and $\mathbf{u}$. Then the hypothesis predicts the $\bar{\mathbf{y}}$ that maximizes $\mathcal{F}$:

$$\bar{\mathbf{y}} = h(\mathbf{x}, \mathbf{u}; \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathcal{F}(\mathbf{x}, \mathbf{u}, \mathbf{y}). \tag{8.2}$$

We describe each $(\mathbf{x}, \mathbf{u}, \mathbf{y})$ through a feature vector $\Psi(\mathbf{x}, \mathbf{u}, \mathbf{y})$; the extraction will be discussed later. The discriminant function $\mathcal{F}(\mathbf{x}, \mathbf{u}, \mathbf{y})$ is assumed to be linear in the feature vector $\Psi(\mathbf{x}, \mathbf{u}, \mathbf{y})$ such that:

$$\mathcal{F}(\mathbf{x}, \mathbf{u}, \mathbf{y}) = \mathbf{w}^{T}\Psi(\mathbf{x}, \mathbf{u}, \mathbf{y}), \tag{8.3}$$

where $\mathbf{w}$ is a weight vector to be learned from training data.

8.2 Structured Learning for Personalized Diversification

In this section, we propose constraints for personalized diversification and describe our optimization problem and the way we make predictions.

8.2.1 Additional constraints

Our personalized diversification model builds on an existing standard structured learning framework (See Section 2.3.7 for details). As discussed above, we aim at training a personalized diversification model that can enforce both diversity and consistency with the user's interest. This can be achieved by introducing additional constraints to the structured SVM optimization problem defined in (2.12). To start, diversity requires a set of retrieved documents that should not discuss the same aspects of an ambiguous query. In other words, aspects of documents returned by a diversification model should have little overlap with one another. Formally, we enforce diversity with the following constraint.

Constraint for diversity:

$$\mathbf{w}^{T}\Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)}) \geq \sum_{y \in \mathbf{y}^{(i)}} \mathbf{w}^{T}\Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, y) - \xi_i. \tag{8.4}$$

In (8.4), the sum of each document's score, $\sum \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, y)$, should not be greater than the overall score when the documents in $\mathbf{y}^{(i)}$ are considered as an ideal ranking of the document sets. As a result, commonly shared features will be associated with relatively low weights, and a document set with less redundancy will be predicted.

Additionally, personalization requires a set of returned documents to match the user's personal interest. Formally, we enforce personalization with the following constraint.

Constraint for consistency with user's interest:

$$\begin{aligned} \mathbf{w}^T\Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)}) \quad &\geq \quad \mathbf{w}^T\Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}) + \\ &\quad (1 - \mathrm{sim}(\mathbf{y}, \mathbf{u}^{(i)})) - \mu - \xi_i, \end{aligned} \tag{8.5}$$

where $\mathrm{sim}(\mathbf{y}, \mathbf{u}^{(i)}) \in [0, 1]$ is a function (see (8.13) below for the definition) that measures subtopic distribution similarity between a set of documents $\mathbf{y}$ and the documents user u_i is interested in, i.e., $\mathbf{u}^{(i)}$, μ is a slack variable that tends to give slightly better performance, which can be defined as $\mu = \frac{1}{N}\sum_{i=1}^{N}(1 - \mathrm{sim}(\mathbf{y}^{(i)}, \mathbf{u}^{(i)}))$.

In (8.5), $(1 - \mathrm{sim}(\mathbf{y}, \mathbf{u}^{(i)}))$ quantifies how well a set of documents matches a user's interest. If the topics discussed in a set of documents $\mathbf{y}$ are not consistent with a user's personal interest, $\mathbf{w}^T\Psi(\mathbf{x}, \mathbf{u}, \mathbf{y})$ will result in a relatively low score. During prediction, documents consistent with a user's interest will be preferred.

8.2.2 Our optimization problem

A set of documents produced in response to an ambiguous query should be diverse and consistent to the user's personal interest. To this end we integrate the proposed additional constraints with standard structured SVMs. We propose to train a personalized diversification model by tackling the following optimization problem:

Optimization Problem 2. (Structured SVMs for personalized diversification)

$$\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{N}\sum_{i=1}^{N}\xi_i \tag{8.6}$$

subject to $\forall i, \forall \mathbf{y} \in \mathcal{Y} \backslash \mathbf{y}^{(i)}, \xi_i \geq 0$,

$$\begin{aligned} \text{i.} \quad & \mathbf{w}^T\Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)}) \quad \geq \quad \mathbf{w}^T\Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}) + \Delta(\mathbf{y}^{(i)}, \mathbf{y}) - \xi_i, \\ \text{ii.} \quad & \mathbf{w}^T\Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)}) \quad \geq \quad \sum_{y \in \mathbf{y}^{(i)}} \mathbf{w}^T\Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, y) - \xi_i, \\ \text{iii.} \quad & \mathbf{w}^T\Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)}) \quad \geq \quad \mathbf{w}^T\Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}) + ((1 - \mathrm{sim}(\mathbf{y}, \mathbf{u}^{(i)})) - \mu) - \xi_i. \end{aligned}$$

8.2.3 The learning algorithm

We can solve the optimization problem defined in (8.6) by employing the cutting plane algorithm (Tsochantaridis et al., 2005). The learning algorithm is shown in Algorithm 4. The algorithm iteratively adds constraints until we have solved the original problem within a desired tolerance ϵ. It starts with empty working sets $\mathcal{W}_i$, $\mathcal{W}_i'$ and $\mathcal{W}_i''$, for $i = 1, \ldots, N$. Then it iteratively finds the most violated constraints $\bar{\mathbf{y}}$, $\bar{\mathbf{y}}'$ and $\mathbf{y}''$ for

Algorithm 4: Cutting plane algorithm

Input : $(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{u}^{(N)}, \mathbf{y}^{(N)}), C, \epsilon$

$\mathcal{W}_i \leftarrow \emptyset, \mathcal{W}_i' \leftarrow \emptyset, \mathcal{W}_i'' \leftarrow \emptyset$ for all $i = 1, \ldots, N$

$\mu = \frac{1}{N} \sum_{i=1}^{N} (1 - \mathrm{sim}(\mathbf{y}^{(i)}, \mathbf{u}^{(i)}))$

repeat

  for $i = 1, \ldots, N$ **do**

    $H(\mathbf{y}; \mathbf{w}) \equiv \Delta(\mathbf{y}^{(i)}, \mathbf{y}) + \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}) - \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)})$

    $H'(\mathbf{y}; \mathbf{w}) \equiv \sum_{y \in \mathbf{y}^{(i)}} \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, y) - \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)})$

    $H''(\mathbf{y}; \mathbf{w}) \equiv \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}) +$

    $((1 - \mathrm{sim}(\mathbf{y}, \mathbf{u}^{(i)})) - \mu) - \mathbf{w}^T \Psi(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)})$

    compute $\bar{\mathbf{y}} = \mathrm{argmax}_{\mathbf{y}} H(\mathbf{y}; \mathbf{w})$, $\bar{\mathbf{y}}' = \mathrm{argmax}_{\mathbf{y}} H'(\mathbf{y}; \mathbf{w})$ and $\bar{\mathbf{y}}'' = \mathrm{argmax}_{\mathbf{y}} H''(\mathbf{y}; \mathbf{w})$

    compute $\xi_i = \max\{0, \max_{\mathbf{y} \in \mathcal{W}_i} H(\mathbf{y}; \mathbf{w}), \max_{\mathbf{y} \in \mathcal{W}_i'} H'(\mathbf{y}; \mathbf{w}), \max_{\mathbf{y} \in \mathcal{W}_i''} H''(\mathbf{y}; \mathbf{w})\}$

    if $H(\bar{\mathbf{y}}; \mathbf{w}) > \xi_i + \epsilon$ **or** $H'(\bar{\mathbf{y}}'; \mathbf{w}) > \xi_i + \epsilon$ **or** $H''(\bar{\mathbf{y}}''; \mathbf{w}) > \xi_i + \epsilon$ **then**

      Add constraint to working set $\mathcal{W}_i \leftarrow \mathcal{W}_i \cup \{\bar{\mathbf{y}}\}$, $\mathcal{W}_i' \leftarrow \mathcal{W}_i' \cup \{\bar{\mathbf{y}}'\}$, $\mathcal{W}_i'' \leftarrow \mathcal{W}_i'' \cup \{\bar{\mathbf{y}}''\}$

      $\mathbf{w} \leftarrow$ optimize (8.6) over $\bigcup_i \{\mathcal{W}_i, \mathcal{W}_i', \mathcal{W}_i''\}$

until *no* $\mathcal{W}_i$*,* $\mathcal{W}_i'$ *and* $\mathcal{W}_i''$ *have changed during iteration*

each $(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}, \mathbf{y}^{(i)})$ in terms of the three constraints (i), (ii) and (iii) in (8.6), respectively. If they are violated by more than ϵ, we add them into the corresponding working sets. We iteratively update $\mathbf{w}$ by optimizing (8.6) over the updated working sets. The outer loop in Algorithm 4 can halt within a polynomial number of iterations for any desired precision ϵ; see (Tsochantaridis et al., 2005).

8.2.4 Prediction

After $\mathbf{w}$ has been learned, given an ambiguous query, a set of candidate documents $\mathbf{x}$, and a set of documents $\mathbf{u}$ the user u is interested in, we try to predict a set of documents $\bar{\mathbf{y}}$ by tackling the following prediction problem:

$$\bar{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathcal{F}(\mathbf{x}, \mathbf{u}, \mathbf{y}) = \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{u}, \mathbf{y}). \tag{8.7}$$

This is a special case of the Budgeted Max Coverage problem (Khuller et al., 1999), and can be efficiently solved by Algorithm 5.

8.3 User-Interest Topic Model and Feature Space

In this section, we first review the notation and terminology used in our user-interest topic model, and then describe the model and the features used in our structured learning framework.

Algorithm 5: Greedy subset selection for prediction

Input : $\mathbf{w}$, $\mathbf{x}$, $\mathbf{u}$

$\bar{\mathbf{y}} \leftarrow \emptyset$

for $k = 1, \ldots, K$ **do**

  $\bar{x} = \arg\max_{x: x \in \mathbf{x}, x \notin \bar{\mathbf{y}}} \mathbf{w}^T \Psi(\mathbf{x}, \mathbf{u}, \bar{\mathbf{y}} \cup \{x\})$

  $\bar{\mathbf{y}} \leftarrow \bar{\mathbf{y}} \cup \{\bar{x}\}$

return $\bar{\mathbf{y}}$

Table 8.1: Additional notation used in the user-interest topic model in this chapter (cf. Table 2.1 and 7.1).

Notation	Gloss
u	user
U	number of users
D	number of documents
$\widetilde{\mathbf{u}}$	a set of users
$\widetilde{\mathbf{w}}$	a set of tokens
b_z	Beta distribution parameter for z
ϑ_u	multinomial distribution of topics specific to u
r_{di}	relevance of the i-th token in d

8.3.1 Notation and terminology

We summarize the main notation used in our user-interest topic model (UIT) in Table 8.1. As we did in Chapter 7, we distinguish between queries, aspects and topics. A *query* is a user's expression of an information need. An *aspect* (sometimes called *subtopic* at the TREC Web tracks (Clarke et al., 2012)) is an interpretation of an information need. We use *topic* to refer to latent topics as identified by a topic modeling method (LDA).

8.3.2 User-interest topic model

To capture per-user and per-document multinomial distributions over topics such that we can measure whether a document can cater for the user's interest, we propose a user-interest latent topic model (UIT). Topic discovery in UIT is influenced not only by token co-occurrences, but also by the relevance scores of documents evaluated by users. In our UIT model, we use a Beta distribution over a (normalized) document relevance span covering all the data, and thus various skewed shapes of rising and falling topic prominence can be flexibly represented.

The latent topic model used in UIT is a generative model of relevance and tokens in the documents. The generative process used in Gibbs sampling (Liu, 1994) for its parameter estimation, is as follows:

i. Draw T multinomials ϕ_z from a Dirichlet prior β, one for each topic z;

ii. For each user u, draw a multinomial ϑ_u from a Dirichlet prior α; then for each token w_{di} in document $d \in \mathbf{u}$:

  (a) Draw a topic z_{di} from multinomial ϑ_u;

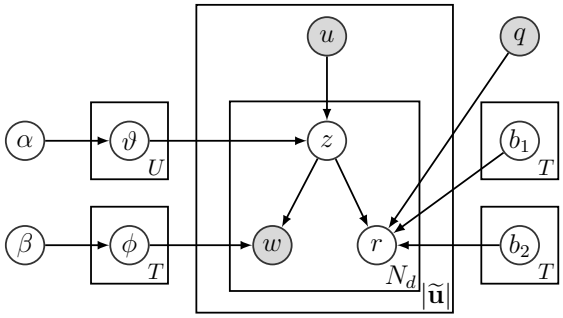


Figure 8.1: Graphical representation of user-interest topic model.

(b) Draw a token w_{di} from multinomial $\phi_{z_{di}}$;

(c) Draw a relevance score r_{di} for w_{di} from Beta $(b_{z_{di}1}, b_{z_{di}2})$.

Fig. 8.1 shows a graphical representation of our model. In the generative process, the relevance scores of tokens observed in the same document are the same and evaluated by a user, although a relevance score is generated for each token from the Beta distribution. In our experiments, there is a fixed number of latent topics, T, although a non-parametric Bayesian version of UIT that automatically integrates over the number of topics would certainly be possible. The posterior distribution of topics depends on the information from two modalities: tokens and the documents' relevance scores.

Inference is intractable in this model. Following (Boyd-Graber and Blei, 2008, 2009; Jameel and Lam, 2013; Liu, 1994; Wang and McCallum, 2006), we employ Gibbs sampling to perform approximate inference. We adopt a conjugate prior (Dirichlet) for the multinomial distributions, and thus we can easily integrate out ϑ and ϕ, analytically capturing the uncertainty associated with them. In this way we facilitate the sampling, i.e., we need not sample ϑ and ϕ at all. Because we use the continuous Beta distribution rather than discretizing document relevance scores, sparsity is not a big concern in fitting the model. For simplicity and speed we estimate these Beta distributions (b_{z1}, b_{z2}) by the method of moments, once per iteration of Gibbs sampling. We find that the sensitivity of the hyper-parameters α and β is not very strong. Thus, for simplicity, we use fixed symmetric Dirichlet distributions ($\alpha = 50/T$ and $\beta = 0.1$) in all our experiments.

In the Gibbs sampling procedure above, we need to calculate the conditional distribution $P(z_{di}|\widetilde{\mathbf{w}}, \mathbf{r}, \mathbf{z}_{-di}, \widetilde{\mathbf{u}}, \alpha, \beta, \mathbf{b}, q)$, where $\mathbf{z}_{-di}$ represents the topic assignments for all tokens except w_{di}. We begin with the joint probability of a dataset, and using the chain rule, we can obtain the conditional probability conveniently as:

$$P(z_{di}|\widetilde{\mathbf{w}}, \mathbf{r}, \mathbf{z}_{-di}, \widetilde{\mathbf{u}}, \alpha, \beta, \mathbf{b}, q) \propto \frac{n_{z_{di}w_{di}} + \beta_{w_{di}} - 1}{\sum_{v=1}^{V}(n_{z_{di}v} + \beta_v) - 1} \frac{n_{u_{di}z_{di}} + \alpha_{z_{di}} - 1}{\sum_{z=1}^{T}(n_{u_{di}z} + \alpha_z) - 1} \times \frac{(1 - r_{di})^{b_{z_{di}1}-1} r_{di}^{b_{z_{di}2}-1}}{B(b_{z_{di}1}, b_{z_{di}2})},$$

where n_{zv} is the total number of tokens v that are assigned to topic z, n_{uz} represents the number of topics z that are assigned to user u. Details are provided in the Appendix 8.A.

After the Gibbs sampling procedure, we can easily infer a user's interest, i.e., multinomial distributions over topics for user u, as:

$$\vartheta_{uz} = p(z|u) = \frac{n_{uz} + \alpha_z}{\sum_{z=1}^{T}(n_{uz} + \alpha_z)}, \tag{8.8}$$

and easily infer multinomial distributions over tokens for topic z:

$$\phi_{zv} = p(v|z) = \frac{n_{zv} + \beta_v}{\sum_{v=1}^{V}(n_{zv} + \beta_v)}, \tag{8.9}$$

where n_{zv} is the number of tokens of word v that are assigned to topic z. To obtain the multinomial distribution over topics for document d, i.e., θ_{dz}, we first apply Bayes' rule:

$$\theta_{dz} = p(z|d) = \frac{p(d|z)p(z)}{p(d)}, \tag{8.10}$$

where $p(d|z)$ is the probability of d belonging to topic z, and $p(z)$ is the probability of topic z. According to (8.9), $p(d|z)$ can be obtained as $p(d|z) = \prod_{v \in d} p(v|z) = \prod_{v \in d} \phi_{zv}$. According to (8.8), $p(z)$ can be obtained as $p(z) = \sum_{u=1}^{U} p(z|u)p(u)$, where U is the total number of users. Therefore, 8.10 can be represented as:

$$\theta_{dz} = \frac{\prod_{v \in d} \phi_{zv} \sum_{u=1}^{U} p(z|u)p(u)}{p(d)}. \tag{8.11}$$

As any d has the same chance to be considered to be returned in response to q, we can assume that $p(d)$ is a constant, and likewise we also assume that $p(u)$ is a constant, such that (8.11) becomes:

$$\theta_{dz} = \tfrac{1}{E} \prod_{v \in d} \phi_{zv} \sum_{u=1}^{U} \vartheta_{uz}, \tag{8.12}$$

where $E = \sum_{z=1}^{T} \prod_{v \in d} \phi_{zv} \sum_{u=1}^{U} \vartheta_{uz}$ is a normalization constant. Then, the topic distribution similarity $\mathrm{sim}(\mathbf{y}, \mathbf{u})$ between a set of documents $\mathbf{y}$ and the documents $\mathbf{u}$ a user u is interested in can be measured as:

$$\mathrm{sim}(\mathbf{y}, \mathbf{u}) = \tfrac{1}{|\mathbf{y}|} \sum_{d \in \mathbf{y}} \cos(\theta_d, \vartheta_u), \tag{8.13}$$

where vectors $\theta_d = (\theta_{d1}, \ldots, \theta_{dT})$ and $\vartheta_u = (\vartheta_{d1}, \ldots, \vartheta_{dT})$ are the multinomial distribution of topics specific to document d and user u, respectively. We use the $\cos$ function in (8.13); other distance functions such as one based on Euclidean distance can be employed but we found that the results were not significantly different.

8.3.3 Feature space

The feature representation Ψ must enable meaningful discrimination between high quality and low quality predictions (Yue and Joachims, 2008). To predict a set of documents in the personalized diversification task, we propose to consider three main types of feature space.

Tokens. Following (Yue and Joachims, 2008), we define L token sets $V_1(\mathbf{y}), \ldots, V_L(\mathbf{y})$. Each token set $V_l(\mathbf{y})$ contains the set of tokens that appear at least l times in some document in $\mathbf{y}$. Then we use thresholds on the ratio $|D_l(v)|/|\mathbf{u}|$ (or $|D_l(v)|/|\mathbf{x}|$) to define feature values of $\psi_l(v, \mathbf{u})$ (or $\psi_l(v, \mathbf{x})$) that describe word v at l-th importance level. Here, $D_l(v)$ is the set of documents that have at least l copies of v in the whole set of documents $\mathbf{u}$ (or $\mathbf{x}$). We let $L = 20$ in our experiments, as quite a few tokens can appear more than 20 times in a document. Besides, we propose to directly utilize the tokens' statistics to capture similarity between a document $x \in \mathbf{y}$ and a set of documents $\mathbf{u}$ that a user u is interested in as features. We consider cosine, Euclidean and Kullback-Leibler (KL) divergence similarity metrics. For each of these three metrics, we compute the minimal, maximal, and average similarity scores of the document $x \in \mathbf{y}$ and the standard deviations to a set of documents $\mathbf{u}$ based on the content of the documents and the standard LDA model (Blei et al., 2003). In total, we have 49 features that fall in this feature category.

Interest. In addition, based on our UIT topic model, we also compute the cosine, Euclidean and KL similarity between a document $x \in \mathbf{y}$ and a set of documents $\mathbf{u}$ based on a multinomial distribution over topics and the user's multinomial distribution over topics generated by UIT. Again, for each of these three similarity metrics, we compute the minimal, maximal, and average similarity scores and the standard deviation scores. In total, we have $S = 36$ features $\omega_s(x, \mathbf{u})$ that fall in this feature category.

Probability. The main probabilities used in state-of-the-art unsupervised personalized diversification methods are utilized in our learning model as features, i.e., $\gamma_m(x, \mathbf{x}, \mathbf{u})$. These probabilities include $p(d|q)$, the probability of d being relevant to q, $p(c|d)$, the probability of d belonging to a category c, $p(c|q, u)$, the personalized query aspect distribution, $p(c|d, u)$, the personalized aspect distribution over d, and $p(d|c, u)$, the personalized aspect-dependent document distribution, where c is a category that d belongs to in the Textwise Open Directory Project category service.[1] For $p(d|q)$, we obtain 3 versions of this feature value produced by BM25 (Robertson and Hull, 2000), Jelinek-Mercer and Dirichlet language models (Zhai and Lafferty, 2001). To get the feature value of $p(c|d)$, we make use of the Textwise service which returns up to 3 possible categories for d, ranked by a score in $[0, 1]$, and we use the normalized scores as features. We adopt 5 ways of computing $p(c|q, u)$ as feature values (Vallet and Castells, 2012); for details on how to compute $p(c|q, u)$, $p(c|d, u)$ and $p(d|c, u)$ we refer to (Vallet and Castells, 2012).

[1] `http://textwise.com`

Then, we define $\Psi(\mathbf{x}, \mathbf{u}, \mathbf{y})$ as follows:

$$\Psi(\mathbf{x}, \mathbf{u}, \mathbf{y}) = \begin{bmatrix} \frac{1}{|\mathbf{y}|} \sum_{v \in V_1(\mathbf{y})} \psi_1(v, \mathbf{u}) \\ \frac{1}{|\mathbf{y}|} \sum_{v \in V_1(\mathbf{y})} \psi_1(v, \mathbf{x}) \\ \vdots \\ \frac{1}{|\mathbf{y}|} \sum_{v \in V_L(\mathbf{y})} \psi_L(v, \mathbf{u}) \\ \frac{1}{|\mathbf{y}|} \sum_{v \in V_L(\mathbf{y})} \psi_L(v, \mathbf{x}) \\ \frac{1}{|\mathbf{y}|} \sum_{x \in \mathbf{y}} \omega_1(x, \mathbf{u}) \\ \vdots \\ \frac{1}{|\mathbf{y}|} \sum_{x \in \mathbf{y}} \omega_S(x, \mathbf{u}) \\ \frac{1}{|\mathbf{y}|} \sum_{x \in \mathbf{y}} \gamma_1(x, \mathbf{x}, \mathbf{u}) \\ \vdots \\ \frac{1}{|\mathbf{y}|} \sum_{x \in \mathbf{y}} \gamma_M(x, \mathbf{x}, \mathbf{u}) \end{bmatrix}.$$

8.4 Experimental Setup

In this section, we describe our experimental setup; Section 8.4.1 lists our research questions; Section 8.4.2 describes our dataset; Section 8.4.3 and Section 8.4.4 lists the baselines and metrics for evaluation, respectively; Section 8.4.5 details the settings of the experiments.

8.4.1 Detailed research questions

We divide our main research question (RQ 5) into the following research questions, and let these questions guide the remainder of the chapter:

RQ 5.1 Can supervised personalized diversification methods outperform state-of-the-art unsupervised methods? Can our method beat state-of-the-art supervised methods? (See Section 8.5.1 for answers.)

RQ 5.2 What is the contribution of the user-interest topic model in our proposed method? (See Section 8.5.2 for the answer.)

RQ 5.3 What is the effect of the constraints for diversity and consistence with user's interest in our method? (See Section 8.5.3 for the answer.)

RQ 5.4 Does our method outperform the best supervised baseline method on each query? (See Section 8.5.4 for the answer.)

RQ 5.5 Can our method retrieve a competitive number of subtopics per query? (See Section 8.5.5 for the answer.)

RQ 5.6 What is the performance of our supervised methods when the C parameter is varied? (See Section 8.5.6 for the answer.)

8.4.2 Dataset

In order to answer our research questions we work with a publicly available personalized diversification dataset.[2] Details of this dataset are discussed in Section 3.2.4. Here we emphasize two important concepts defined in the dataset–*user relevance* and *topic relevance*: each relevance judgement includes 3 main assessments: a 4-grade scale assessment on how relevant the result is to the user's interests (resulting in the *user relevance* ground truth and the set of users' interesting documents being created); a 4-grade scale assessment on how relevant the result is to the evaluated query (resulting in the *topic relevance* ground truth being created); and a 2-grade assessment whether a specific subtopic is related to the evaluated query (resulting in the subjective subtopics related to the search query being created). For pre-processing, we apply Porter stemming, tokenization, and stopword removal (using the INQUERY list) to the documents using the Lemur toolkit.[3]

Two well-known corpora, ClueWeb09 and ClueWeb12,[4] have been proposed for search result diversification tasks in the TREC 2009–2013 Web tracks (Clarke et al., 2012). However, they do not contain any user information or relevance judgments provided by specific users, and thus do not fit our experiments.

8.4.3 Baselines

Let PSVM$_{div}$ denote our personalized diversification via structured learning method. We compare PSVM$_{div}$ to eleven baselines: a traditional web search algorithm, BM25 (Robertson and Hull, 2000); two well-known plain (in the sense of "not personalized") search result diversification approaches, IA-Select (Agrawal et al., 2009) and xQuAD (Santos et al., 2010a); a plain (in the sense of "not diversified") personalized search approach based on BM25 (Vallet et al., 2010), Pers$_{BM25}$; a two-stage diversification and personalization approach, xQuAD$_{BM25}$, as suggested by (Radlinski and Dumais, 2006), that first applies the xQuAD algorithm and then Pers$_{BM25}$; 4 state-of-the-art unsupervised personalized diversification methods (Vallet and Castells, 2012), PIA-Select, PIA-Select$_{BM25}$, PxQuAD, and PxQuAD$_{BM25}$. As PSVM$_{div}$ builds on standard structured learning framework, we also consider 2 structured learning algorithms: SVM$_{div}$ (Yue and Joachims, 2008) that directly tries to retrieve relevant documents covering as many subtopics as possible, and a standard structured learning method, denoted as SVM$_{rank}$ (Yue et al., 2007) that directly ranks documents by optimizing a relevance-biased evaluation metric (we use α-nDCG and nDCG to define the loss functions for SVM$_{div}$ and SVM$_{rank}$, respectively).[5]

For the supervised methods, PSVM$_{div}$, SVM$_{div}$ and SVM$_{rank}$, we use a 130/40/10 split for our training, validation and test sets, respectively. We train PSVM$_{div}$, SVM$_{div}$ and SVM$_{rank}$ using values of C (see (8.6)) that vary from 1e-4 to 1.0. The best C value is then chosen on the validation set, and evaluated on the test queries. The train/validation/test splits are permuted until all 180 queries were chosen once for the test set. We repeat the experiments 10 times and report the average evaluation results.

[2] http://ir.ii.uam.es/~david/persdivers/

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

[4] http://boston.lti.cs.cmu.edu/clueweb12/

[5] The source code for SVM$_{rank}$ (Yue et al., 2007) and SVM$_{div}$ (Yue and Joachims, 2008) is available at http://www.cs.cornell.edu/People/tj/.

8.4.4 Evaluation

We use the following diversity metrics for evaluation, most of which are official evaluation metrics in the TREC Web tracks (Clarke et al., 2012) and are widely used in the literature on result diversification: α-nDCG@k, S-Recall@k, ERR-IA@k, Prec-IA@k and MAP-IA@k. See Section 3.3.3 on how to compute evaluation scores in terms of these metrics.

For evaluating accuracy, we use nDCG (Järvelin and Kekäläinen, 2002), ERR, Prec@k and MAP. Also, see Section 3.3.1 on how to compute evaluation scores in terms of these metrics. Since users mainly evaluated the top 5 returned results (Vallet and Castells, 2012), we compute the scores at depth 5 for all metrics.

8.4.5 Experiments

We report on 6 main experiments aimed at answering the research questions listed in Section 8.4.1. Our first experiment aims at understanding whether supervised personalized diversification methods outperform unsupervised ones and whether PSVM$_{div}$ beats the supervised algorithms that apply structured learning technique directly. We compare PSVM$_{div}$ to two supervised baselines, SVM$_{div}$ and SVM$_{rank}$, and the nine unsupervised baselines with both topic relevance and user relevance ground truths, respectively.

To understand the contribution of the user-interest topic model, we conduct our second experiment where we perform comparisons between PSVM$_{div}$ using all features ("token", "interest" and "probability," see Section 8.3.3) including those extracted from the topic model and PSVM$_{div}$ using basic features ("token" and "probability" only, see Section 8.3.3). In our third experiment, aimed at understanding the effect of our new constraints in PSVM$_{div}$, a series of experiments is conducted by employing different sets of constraints while training.

In order to understand how PSVM$_{div}$ compares to the best baseline, our fourth and fifth experiment provide a query- and subtopic-level analysis, respectively. Finally, to understand the influence of the key parameter in our structured learning framework, C, we train PSVM$_{div}$, SVM$_{div}$ and SVM$_{rank}$ by varying C from 1e-4 to 1.0 and report the performance.

8.5 Results and Analysis

The following sections report, analyze and discuss our experimental results.

8.5.1 Supervised vs. unsupervised

Table 8.2 lists the diversity scores of the unsupervised baseline methods. For all metrics in terms of either user relevance or topic relevance, none of the plain methods, viz., BM25, IA-Select,Pers$_{BM25}$, xQuAD and xQuAD$_{BM25}$, beats the best unsupervised personalized diversification methods, viz., PIA-Select, PIA-Se-lect$_{BM25}$, PxQuAD or PxQuAD$_{BM25}$. Moreover, in some cases the performance differences between the best plain method and the best unsupervised personalized diversification method are significant. This indicates that diversity and personalization are complementary and can en-

hance each other. The same observation can be found in Table 8.5 where performance is evaluated by relevance-oriented metrics.

Table 8.3 shows the diversity-oriented evaluation results of three supervised methods using basic features ("token", and "probability" features, see Section 8.3.3) in terms of both ground truths. In terms of diversity-oriented evaluation metrics all of the supervised methods significantly outperform the best unsupervised methods when making comparisons between the scores and the scores of unsupervised methods in Table 8.2 in most cases. We make further comparisons in Tables 8.5 and 8.6 in terms of relevance-oriented metrics, and find that supervised methods can statistically significantly outperform unsupervised ones. These two findings attest to the merits of taking supervised personalized diversification methods for the task of personalized search result diversification.

Next, we compare supervised strategies to each other. Tables 8.3 and 8.4 show the diversity-oriented evaluation resutls in terms of both grounth truths. It is clear from both tables that our supervised method PSVM$_{div}$ statistically significantly beats plain supervised methods, SVM$_{rank}$ and SVM$_{div}$. This is because PSVM$_{div}$ considers both personalization and diversity factors, whereas the other two do not take both two factors into account. SVM$_{rank}$ only tries to return more relevant documents, and SVM$_{div}$ directly utilizes standard structured learning for diversification.

As shown in Table 8.6, in terms of the relevance-oriented metrics, PSVM$_{div}$ does not significantly outperform SVM$_{rank}$ and SVM$_{div}$. This is because PSVM$_{div}$ returns the same number of relevant documents, however, cover more subtopics than the other supervised methods. Hence, PSVM$_{div}$ mainly outperforms the other two in terms of diversity-oriented metrics. We provide further analyses in Section 8.5.4 (query-level) and Section 8.5.5 (subtopic-level).

8.5.2 Effect of the proposed UIT model

Next, to understand the contribution of our UIT topic model, we compare the performance of the supervised methods using basic features, i.e., all other features but not the features generated from the UIT model, with those using all the features.

We turn to Tables 8.3 and 8.4, that list the results of the supervised methods in terms of diversity-oriented metrics when using the basic features and all features, respectively. For all supervised methods, the performance of using all features is better than that of only using the basic features. That is, our proposed UIT model can capture users' interest distributions and this kind of information can be applied to improve performance. Due to space limitations, we do not report the results in terms of relevance-oriented metrics; the findings there are qualitatively similar.

8.5.3 Effect of the proposed constraints

Next, to understand the effect of the newly proposed constraints, we conduct experiments by employing different sets of constraints while training. The comparisons are again divided into those using all features and those using basic features. We write PSVM$_{div}$-C$_{\text{i}}$, PSVM$_{div}$-C$_{\text{i,ii}}$, PSVM$_{div}$-C$_{\text{i,iii}}$, and PSVM$_{div}$-All to denote the methods trained with the standard constraint (constraint i in (8.6)), standard and diversity-biased constraints (constraints i and ii in (8.6)), standard and interest-biased (constraints i and iii in (8.6)),

Table 8.2: Performance of unsupervised methods on diversification metrics. The best performance per metric is in boldface. The best plain retrieval method (BM25, IA-Select, Pers$_{BM25}$, xQuAD and xQuAD$_{BM25}$) is underlined. Statistically significant differences between the best performance per metric and the best plain retrieval method are marked in the upper left hand corner of the best performance score.

	User relevance				
	α-nDCG	S-Recall	ERR-IA	Prec-IA	MAP-IA
BM25	.6443	.4557	.2267	.1659	.1245
IA-Select	.6099	.4282	.2241	.1624	.1177
Pers$_{BM25}$	.6427	.4541	.2318	.1639	.1206
xQuAD	.6421	.4635	.2299	.1675	.1267
xQuAD$_{BM25}$	.6270	.4558	.2249	.1646	.1123
PIA-Select	.5766	.4407	.2006	.1480	.1085
PIA-Select$_{BM25}$	.6457	▲**.4752**	.2364	.1581	.1180
PxQuAD	.6409	.4588	.2313	.1629	.1296
PxQuAD$_{BM25}$	**.6497**	.4713	△**.2367**	**.1676**	**.1296**
	Topic relevance				
BM25	.7599	.4456	.2315	.1717	.1241
IA-Select	.7685	.4425	.2365	.1767	.1212
Pers$_{BM25}$	.7746	.4555	.2330	.1794	.1219
xQuAD	.7711	.4600	.2348	.1747	.1245
xQuAD$_{BM25}$	.7763	.4741	.2336	.1773	.1225
PIA-Select	.7410	.4641	.2227	.1650	.1206
PIA-Select$_{BM25}$	△**.7854**	**.4798**	△**.2415**	.1740	▲**.1300**
PxQuAD	.7744	.4543	.2350	.1747	.1278
PxQuAD$_{BM25}$	.7827	.4718	.2396	**.1797**	.1245

and all constraints involved (constraints i, ii and iii in (8.6)), respectively. Again, we only report results on diversity-oriented metrics.

According to Tables 8.7 and 8.8, when employing one more constraint, either diversity-biased or interest-biased, the performance is statistically significantly better than that of only employing the standard constraint. In terms of all metrics, the performance of PSVM$_{div}$ employing all constraints statistically significantly outperforms the performance of using at most two constraints. The positive effect of the proposed constraints again demonstrates that combining diversification (the diversity-biased constraint) and personalization (the interest-biased constraint) boosts the performance.

8.5.4 Query-level analysis

In order to figure out why PSVM$_{div}$ improves over other supervised baselines, we take a closer look at per test query improvements of PSVM$_{div}$ over the best supervised baseline method, viz., SVM$_{div}$, which outperforms SVM$_{rank}$ in most cases. Fig. 8.2 shows the

Table 8.3: Performance of supervised methods utilizing basic features on diversification metrics. The best performance per metric is in boldface. Statistically significant differences between supervised and the best unsupervised method (in Table 8.2) per metric, between PSVM$_{div}$ and SVM$_{div}$, are marked in the upper left hand corner of the supervised method' score, in the right hand corner of the PSVM$_{div}$ score, respectively.

	User relevance				
	α-nDCG	S-Recall	ERR-IA	Prec-IA	MAP-IA
SVM$_{rank}$	▲.6667	△.4837	.2396	.1683	▲.1856
SVM$_{div}$	▲.6750	△.4887	.2412	△.1698	▲.1974
PSVM$_{div}$	▲**.7234**▲	▲**.5756**▲	▲**.2514**▲	▲**.1702**△	▲**.2037**△
	Topic relevance				
SVM$_{rank}$	.7889	.4805	.2437	△.1812	▲.1848
SVM$_{div}$	▲.8003	△.4893	△.2479	△.1833	▲.2045
PSVM$_{div}$	▲**.8533**▲	▲**.5834**▲	▲**.2649**▲	▲**.1846**△	▲**.2113**▲

Table 8.4: Performance of supervised methods utilizing all features on diversification metrics. The best performance per metric is in boldface. All the scores here are statistically significant compared to those in Table 8.2. Statistically significant differences between the method here and the method in Table 8.3, between PSVM$_{div}$ and SVM$_{div}$, are marked in the upper left hand corner of the corresponding score, in the right hand corner of the PSVM$_{div}$ score, respectively.

	User relevance				
	α-nDCG	S-Recall	ERR-IA	Prec-IA	MAP-IA
SVM$_{rank}$	△.6782	△.4973	.2416	△.1710	▲.2887
SVM$_{div}$	△.6867	△.4973	.2456	△.1729	▲.2911
PSVM$_{div}$	▲**.7513**▲	▲**.6140**▲	△**.2628**▲	△**.1742**△	▲**.2979**△
	Topic relevance				
SVM$_{rank}$	▲.8422	▲.5068	▲.2554	△.1903	▲.3001
SVM$_{div}$	△.8569	▲.5068	▲.2628	▲.1907	▲.3036
PSVM$_{div}$	▲**.9549**▲	▲**.6730**▲	▲**.2849**▲	△**.1917**△	▲**.3096**△

per query performance differences in terms of the diversify-oriented metrics of PSVM$_{div}$ against SVM$_{div}$ when they use all the features. PSVM$_{div}$ achieves performance improvements for many queries, especially in terms of α-nDCG, S-Recall, ERR-IA.

In a very small number of cases, PSVM$_{div}$ performs poorer than SVM$_{div}$. This appears to be due to the fact that PSVM$_{div}$ promotes some non-relevant documents when it tries to cover as many subtopics as possible for a given query.

Table 8.5: Performance of unsupervised methods on relevance metrics. Notational conventions are the same as in Table 8.2.

	User relevance				Topic relevance			
	nDCG	ERR	Prec	MAP	nDCG	ERR	Prec	MAP
BM25	.5697	.9364	.7113	.2038	.7775	.9440	.9146	.2239
IA-Select	.5126	.9389	.6796	.1813	.7340	.9452	.9250	.2299
Pers$_{BM25}$	.5713	.9276	.7183	.2076	.7741	.9374	.9298	.2316
xQuAD	.5526	.9352	.6858	.1915	.7518	.9367	.9125	.2231
xQuAD$_{BM25}$	.5540	.9133	.6921	.1841	.7605	.9278	.9312	.2281
PIA-Select	.4783	.9034	.6417	.1774	.6709	.9062	.8667	.2043
PIA-Select$_{BM25}$	.5482	.9271	.6687	.1803	.7264	.9418	.9042	.2223
PxQuAD	.5631	.9246	.7050	.2073	.7679	.9435	.9229	.2306
PxQuAD$_{BM25}$	**.5764**	**.9374**	$^{\triangle}$**.7258**	$^{\triangle}$**.2145**	**.7793**	**.9466**	**.9396**	**.2355**

Table 8.6: Performance of supervised methods utilizing basic features on relevance metrics. The best performance per metric is in boldface. Statistically significant differences between supervised and the best unsupervised method (in Table 8.5) per metric, between PSVM$_{div}$ and SVM$_{div}$, are marked in the upper left hand corner of the supervised method' score, in the right hand corner of the PSVM$_{div}$ score, respectively.

	User relevance				Topic relevance			
	nDCG	ERR	Prec	MAP	nDCG	ERR	Prec	MAP
SVM$_{rank}$	$^{\triangle}$.5805	$^{\triangle}$.9456	$^{\triangle}$.7345	$^{\triangle}$.2238	$^{\triangle}$.7864	.9478	$^{\blacktriangle}$.9763	$^{\triangle}$.2446
SVM$_{div}$	$^{\triangle}$.5813	$^{\triangle}$.9467	$^{\triangle}$.7396	$^{\triangle}$.2240	$^{\triangle}$.7858	.9493	$^{\blacktriangle}$.9806	$^{\triangle}$.2482
PSVM$_{div}$	$^{\triangle}$**.5833**	$^{\triangle}$**.9485**	$^{\triangle}$**.7412**	$^{\triangle}$**.2281**	$^{\triangle}$**.7922**$^{\triangle}$	$^{\triangle}$**.9521**	$^{\blacktriangle}$**.9834**	$^{\triangle}$**.2496**

Table 8.7: Performance of PSVM$_{div}$ involving different constraints using basic features on diversification metrics with user relevance ground truth. The best performance per metric is in boldface. Statistically significant differences against PSVM$_{div}$-C$_{\text{i}}$ are marked in the upper right hand corner of the corresponding scores.

	User relevance				
	α-nDCG	S-Recall	ERR-IA	Prec-IA	MAP-IA
PSVM$_{div}$-C$_{\text{i}}$	.6713	.4842	.2403	.1673	.1969
PSVM$_{div}$-C$_{\text{i,ii}}$	.6973$^{\blacktriangle}$	.5262$^{\blacktriangle}$	.2437	.1681	.1977
PSVM$_{div}$-C$_{\text{i,iii}}$	.6994$^{\blacktriangle}$	.5275$^{\blacktriangle}$	.2478$^{\triangle}$	.1687$^{\triangle}$	.1983
PSVM$_{div}$-All	**.7234**$^{\blacktriangle}$	**.5756**$^{\blacktriangle}$	**.2514**$^{\blacktriangle}$	**.1702**$^{\blacktriangle}$	**.2037**$^{\triangle}$

8.5.5 Subtopic-level analysis

Next, we zoom in on the number of different subtopics that are returned by PSVM$_{div}$ and SVM$_{div}$, respectively, to further analyze why PSVM$_{div}$ beats SVM$_{div}$. Here, again,

Table 8.8: Performance of PSVM$_{div}$ involving different constraints using all features on diversification metrics with user relevance ground truth. Statistically significant differeneces between the score here and that in Table 8.7 are marked in the upper left hand corner of the scores. Other notational conventions are the same as in Table 8.7.

	User relevance				
	α-nDCG	S-Recall	ERR-IA	Prec-IA	MAP-IA
PSVM$_{div}$-C$_{i}$	▲.6843	▲.4965	△.2434	△.1714	▲.2906
PSVM$_{div}$-C$_{i,ii}$	▲.7156▲	▲.5334▲	△.2494△	△.1720△	▲.2932
PSVM$_{div}$-C$_{i,iii}$	▲.7194▲	▲.5388▲	△.2501△	△.1723△	▲.2937△
PSVM$_{div}$-All	▲**.7513**▲	▲**.6140**▲	△**.2628**▲	△**.1742**▲	▲**.2979**△

we use SVM$_{div}$ as a representative. Specifically, we report changes in the number of subtopics for PSVM$_{div}$ against SVM$_{div}$ in Fig. 8.3 when they use all features. Red bars indicate the number of subtopics that appear in the run of PSVM$_{div}$ but not in the run of SVM$_{div}$, white bars indicate the number of subtopics in both runs, whereas blue bars indicate the number of subtopics that are not in PSVM$_{div}$ but in SVM$_{div}$; queries are ordered first by the size of the red bar, then the size of the white bar, and finally the size of the blue bar.

Clearly, the differences between PSVM$_{div}$ and SVM$_{div}$ in the top 2 and 3 are more limited than the differences in the top 4 and 5, but in all cases PSVM$_{div}$ outperforms SVM$_{div}$. E.g., in total there are 68 more subtopics in the top 5 of the run produced by PSVM$_{div}$ than those in the SVM$_{div}$ run (in terms of all the 180 test queries, 68 subtopics in PSVM$_{div}$ but not in SVM$_{div}$, 7 subtopics in SVM$_{div}$ but not in PSVM$_{div}$).

8.5.6 Performance of parameter tuning

To understand the performance of the tradeoff parameter C used in (2.12) (see Section 2.3.7) and (8.6), which balances between weights and slacks, we show the performance of PSVM$_{div}$ as well as the two supervised baselines using all features. To save space, we only report the performance on α-nDCG. Fig. 8.4 plots the results and it illustrates that PSVM$_{div}$ performs best when C is small. This indicates the merit of our new constraints (as well as the standard constraint used in the baselines) focusing on weight modification rather than on low training loss.

8.6 Conclusion

Most previous work on personalized diversification of search results produce a ranking using unsupervised methods, either implicitly or explicitly. In this chapter, we have adopted a different perspective on the problem, based on structured learning. We propose to boost the diversity and match to users' personal interests of search results by introducing two additional constraints into the standard structured learning framework. We also propose a user-interest topic model to capture users' multinomial distribution of interest

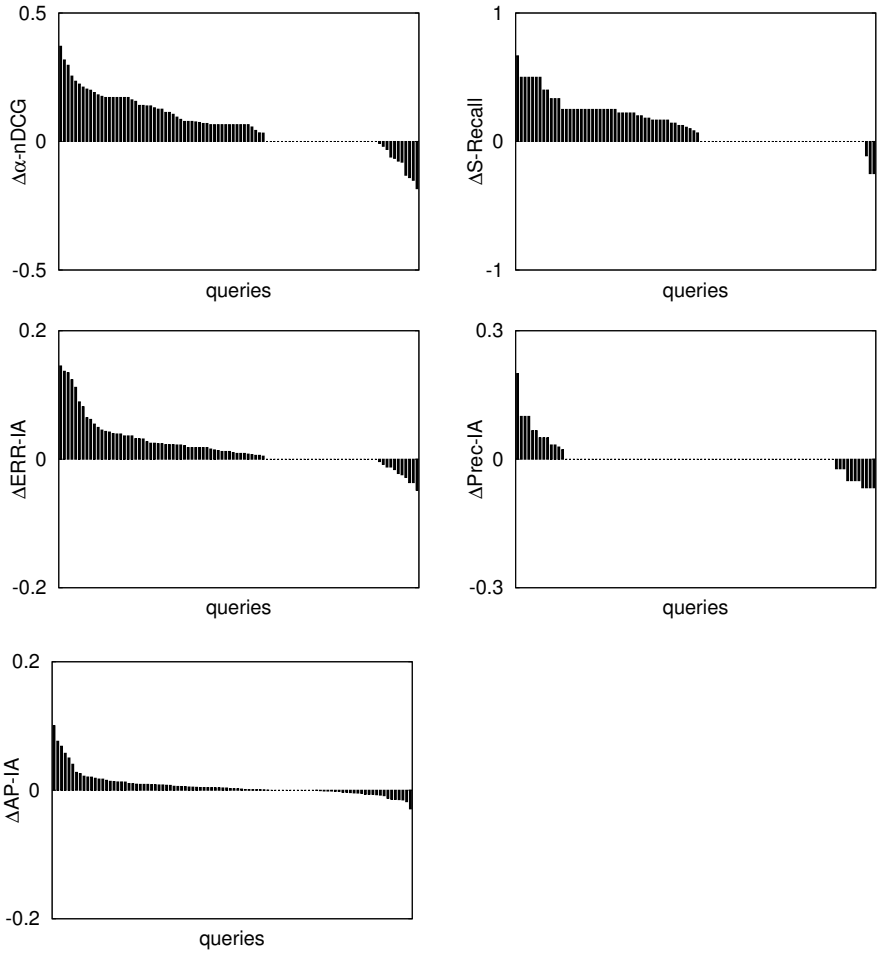


Figure 8.2: Per query performance differences of PSVM$_{div}$ against SVM$_{div}$. The figures shown are for α-nDCG, S-Recall, ERR-IA, Prec-IA and MAP-IA, respectively. A bar extending above the center of a plot indicates that PSVM$_{div}$ outperforms SVM$_{div}$, and vice versa for bars extending below the center. Note that figures are not in the same scale.

over topics and infer per-document multinomial distributions over topics. Based on this a number of user interest features are extracted and the similarity between a user and a document can be effectively measured for our learning method. Our experiments provide answers to the main research question raised at the beginning of this chapter:

RQ 5 How to enhance both diversification and personalization performance at the same time in a supervised way?

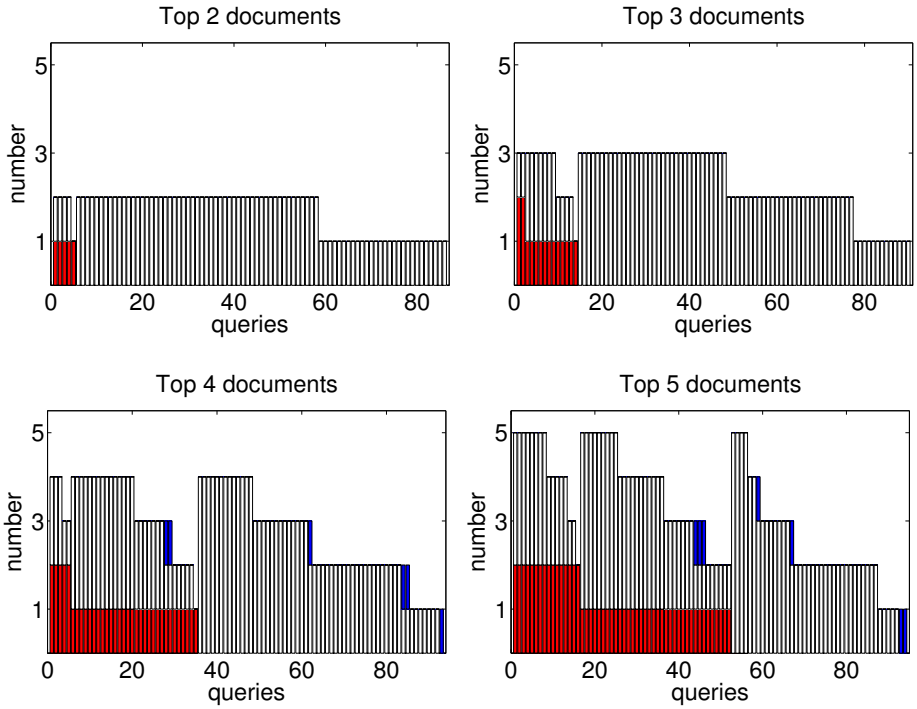


Figure 8.3: How runs produced by PSVM$_{div}$ and SVM$_{div}$ differ. Red, white, blue bars indicate the number of different subtopics that appear in PSVM$_{div}$ but not in SVM$_{div}$, in both runs and not in PSVM$_{div}$ but in SVM$_{div}$, respectively, at corresponding depth k (for k=2, 3, 4, 5). Figures should be viewed in color.

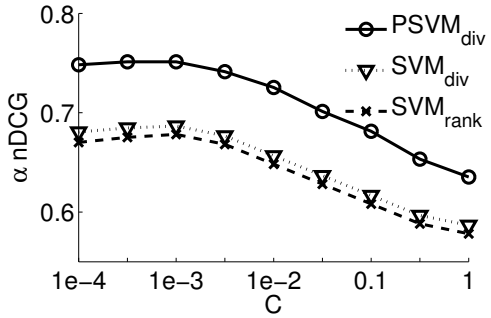


Figure 8.4: Performance of the supervised methods using all features when varying the value of parameter C.

To answer the main research question, we worked with a publicly available personalized diversification dataset. Our evaluation in the experiments showed that supervised personalized diversification approaches outperforms state-of-the-art unsupervised personalization diversification, plain personalization and plain diversification algorithms. The two

proposed constraints are shown to play a significant role in the supervised method. We also found that the user-interest topic model helps to improve performance. In addition, our proposed learning method is able to return more subtopics.

As to future work, we aim to study other types of learning strategies for personalized diversification of search results. Our method employed the α-nDCG metric in the loss function; we plan to use other alternative metrics. Finally, our experimental results were only evaluated on a single dataset. In future work we plan to invite users to label the existing datasets, e.g., ClueWeb09, such that they can also be used for personalized diversification algorithms.

This chapter is the last research chapter of the thesis. The next chapter is used to summarize the research presented in the thesis, to answer the research questions, and to give directions for future research based on findings in this thesis.

8.A Gibbs Sampling Derivation for UIT Model

We begin with the joint distribution $P(\widetilde{\mathbf{w}}, \mathbf{r}, \mathbf{z}, \widetilde{\mathbf{u}}|\alpha, \beta, \mathbf{b}, q)$. We can take advantage of conjugate priors to simplify the integrals. All symbols are defined in Sections 8.1, 8.2 and 8.3.

$$
\begin{aligned}
&P(\widetilde{\mathbf{w}}, \mathbf{r}, \mathbf{z}, \widetilde{\mathbf{u}}|\alpha, \beta, \mathbf{b}, q) = P(\widetilde{\mathbf{w}}|\mathbf{z}, \beta)p(\mathbf{r}|\mathbf{b}, \mathbf{z}, q)P(\mathbf{z}|\widetilde{\mathbf{u}}, \alpha) \\
&= \int P(\widetilde{\mathbf{w}}|\mathbf{\Phi}, \mathbf{z})p(\mathbf{\Phi}|\beta)d\mathbf{\Phi} \times p(\mathbf{r}|\mathbf{b}, \mathbf{z}, q) \int P(\mathbf{z}|\widetilde{\mathbf{u}}, \mathbf{\Theta})p(\mathbf{\Theta}|\alpha)d\mathbf{\Theta} \\
&= \int \prod_{d=1}^{D}\prod_{i=1}^{N_d} P(w_{di}|\phi_{z_{di}}) \prod_{z=1}^{T} p(\phi_z|\beta)d\mathbf{\Phi} \\
&\quad \times \prod_{d=1}^{D}\prod_{i=1}^{N_d} p(r_{di}|b_{z_{di}1}, b_{z_{di}2}, q) \\
&\quad \times \int \prod_{d=1}^{D}\prod_{i=1}^{N_d} P(z_{di}|\vartheta_u) \prod_{u=1}^{U} p(\vartheta_u|\alpha)d\mathbf{\Theta} \\
&= \int \prod_{z=1}^{T}\prod_{v=1}^{V} \phi_{z_v}^{n_{z_v}} \prod_{z=1}^{T} \left(\frac{\Gamma(\sum_{v=1}^{V}\beta_v)}{\prod_{v=1}^{V}\Gamma(\beta_v)} \prod_{v=1}^{V} \phi_{z_v}^{\beta_v - 1} \right) d\mathbf{\Phi} \\
&\quad \times \prod_{d=1}^{D}\prod_{i=1}^{N_d} p(r_{di}|b_{z_{di}1}, b_{z_{di}2}, q) \\
&\quad \times \int \prod_{u=1}^{U}\prod_{z=1}^{T} \vartheta_{u_z}^{n_{u_z}} \prod_{u=1}^{U} \left(\frac{\Gamma(\sum_{z=1}^{T}\alpha_z)}{\prod_{z=1}^{T}\Gamma(\alpha_z)} \prod_{z=1}^{T} \vartheta_{u_z}^{\alpha_z - 1} \right) d\mathbf{\Theta} \\
&= \left(\frac{\Gamma(\sum_{v=1}^{V}\beta_v)}{\prod_{v=1}^{V}\Gamma(\beta_v)} \right)^{T} \left(\frac{\Gamma(\sum_{z=1}^{T}\alpha_z)}{\prod_{z=1}^{T}\Gamma(\alpha_z)} \right)^{U} \\
&\quad \times \prod_{d=1}^{D}\prod_{i=1}^{N_d} p(r_{di}|b_{z_{di}1}, b_{z_{di}2}, q) \\
&\quad \times \prod_{z=1}^{T} \frac{\prod_{v=1}^{V}\Gamma(n_{z_v} + \beta_v)}{\Gamma(\sum_{v=1}^{V}(n_{z_v} + \beta_v))} \prod_{u=1}^{U} \frac{\prod_{z=1}^{T}\Gamma(n_{u_z} + \alpha_z)}{\Gamma(\sum_{z=1}^{T}(n_{u_z} + \alpha_z))}
\end{aligned}
$$

Applying the chain rule, we can obtain the conditional probability:

$$
\begin{aligned}
&P(z_{di}|\widetilde{\mathbf{w}}, \mathbf{r}, \mathbf{z}_{-di}, \widetilde{\mathbf{u}}, \alpha, \beta, \mathbf{b}, q) \\
&= \frac{P(z_{di}, w_{di}, r_{di}, u_{di}|\widetilde{\mathbf{w}}_{-di}, \mathbf{r}_{-di}, \mathbf{z}_{-di}, \widetilde{\mathbf{u}}_{-di}, \alpha, \beta, \mathbf{b}, q)}{P(w_{di}, r_{di}, u_{di}|\widetilde{\mathbf{w}}_{-di}, \mathbf{r}_{-di}, \mathbf{z}_{-di}, \widetilde{\mathbf{u}}_{-di}, \alpha, \beta, \mathbf{b}, q)} \\
&= \frac{P(\widetilde{\mathbf{w}}, \mathbf{r}, \mathbf{z}, \widetilde{\mathbf{u}}|\alpha, \beta, \mathbf{b}, q)}{P(\widetilde{\mathbf{w}}, \mathbf{r}, \mathbf{z}_{-di}, \widetilde{\mathbf{u}}|\alpha, \beta, \mathbf{b}, q)} \\
&\text{because } z_{di} \text{ depends only on } w_{di}, r_{di} \text{ and } u_{di} \\
&\propto \frac{P(\widetilde{\mathbf{w}}, \mathbf{r}, \mathbf{z}, \widetilde{\mathbf{u}}|\alpha, \beta, \mathbf{b}, q)}{P(\widetilde{\mathbf{w}}_{-di}, \mathbf{r}_{-di}, \mathbf{z}_{-di}, \widetilde{\mathbf{u}}_{-di}|\alpha, \beta, \mathbf{b}, q)} \\
&\propto \frac{n_{z_{di}w_{di}} + \beta_{w_{di}} - 1}{\sum_{v=1}^{V}(n_{z_{di}v} + \beta_v) - 1} \frac{n_{u_{di}z_{di}} + \alpha_{z_{di}} - 1}{\sum_{z=1}^{T}(n_{u_{di}z} + \alpha_z) - 1}
\end{aligned}
$$

$$\times \frac{(1-r_{di})^{b_{z_{di}1}-1} r_{di}^{b_{z_{di}2}-1}}{B(b_{z_{di}1}, b_{z_{di}2})}$$

As relevance is drawn from continuous Beta distributions, sparsity is not a big problem for parameter estimation of $\mathbf{b}$. For simplicity, we update $\mathbf{b}$ after each Gibbs sample by the method of moments as:

$$b_{z1} = \bar{t}_z \left(\frac{\bar{t}_z(1-\bar{t}_z)}{s_z^2} - 1 \right),$$
$$b_{z2} = (1-\bar{t}_z) \left(\frac{\bar{t}_z(1-\bar{t}_z)}{s_z^2} - 1 \right),$$

where $\bar{t}_z$ and s_z^2 are the sample mean and biased sample variance of the relevance belonging to topic z, respectively.

9 Conclusions

The research presented in this thesis revolves around two research themes: How to improve the performance of data fusion and search result diversification in information retrieval. The five research chapters of this thesis address the challenges of data fusion and search result diversification in IR as follows. First, in Chapter 4, we focused on how to employ data fusion to enhance retrieval performance of microblog search. In particular, we developed a novel probabilistic data fusion model, BurstFuseX, that is burst-aware and not only utilizes information traditionally used when merging ranked lists, but also exploits temporal information of the microblog posts. Second, in Chapter 5, we revisited the problem of utilizing data fusion approaches for microblog search. We still focused on how to merge result lists for microblog search, but also inferred the rank scores of missing documents via latent factor modeling in data fusion. Third, in Chapter 6, we turned to the problem of data fusion for ad hoc search, and experimentally verified the fusion performance via a manifold-based algorithm. In Chapter 7, we examined the hypothesis that data fusion can improve performance in terms of diversity metrics, and introduced a diversified data fusion method that infers latent topics of an ambiguous query when fusing result lists. Finally, in Chapter 8, we focused on the problem of personalized search result diversification, and proposed a structured learning framework to deal with the problem.

Below, we provide a more detailed summary of the contributions and results of our research, and answer the research questions set out at the beginning of the thesis (Section 9.1). We conclude with an outlook on future research directions (Section 9.2).

9.1 Main Findings

The goal that we have addressed in this thesis is to improve the performance of data fusion and search result diversification in IR. We began the research part of this thesis by focusing on employing data fusion approaches for post search in microblogging environments. In particular, the research questions we addressed in Chapter 4 focused on how to utilize data fusion approaches to tackle the problem of microblog search:

RQ 1 Can data fusion help microblog search?

To answer this question, we have focused on utilizing a special feature for boosting the performance of search algorithms for microblog posts. We have proposed a data fusion

approach, BurstFuseX that fuses result lists based in part on the bursty nature of many discussions on microblog platforms. BurstFuseX not only utilizes ranking information of microblog posts but also exploits their timestamps. Specifically, our approach is based on integrating information generated by a standard fusion method, such as CombSUM, CombMNZ or λ-Merge, detecting bursts of posts across the lists being fused, and rewarding posts that are published in or near a burst containing highly ranked posts.

Using BurstFuseX, we analyzed the search performance in a microblogging environment and showed that detecting bursts and then using burst information together with a standard fusion method can enhance the retrieval performance compared to the standard fusion method it integrates, in terms of mean average precision as well as precision-oriented measures. Our new fusion method has a strong recall-enhancing effect; compared to the standard fusion method it incorporates, this comes at a small price in terms of a small drop in very early precision measures such as p@5. Our findings also showed that our BurstFuseX method can significantly outperform burst or time-sensitive retrieval models and models that detect bursts based on the content of posts.

In previous data fusion approaches, documents in the fused list are ranked in decreasing order of their fusion scores. The fusion score of a document is usually the sum of rank scores from the individual input lists. Previous work on data fusion often assumes, either implicitly or explicitly, that the rank score of a document is set to zero if the document does not appear in the input list. We revisited the problem of searching posts using data fusion and challenged this assumption, resulting in the following questions:

RQ 2 How to infer scores of so-called missing documents in data fusion?

We focused on utilizing time information to boost the performance of searching microblog posts. Specifically, we proposed a novel rank aggregation approach, TimeRA, that utilizes bursts and only rank information to fuse result lists. TimeRA first detects bursts of posts across the lists utilizing the original rank information of the posts, and then rewards posts that are ranked low in a few lists but in the vicinity of a burst that contains higher ranked posts. It also infers the rank scores of so-called missing posts by modeling lists and posts as a mixture of latent factors.

We experimentally showed that both utilizing burst information and score inference for data fusion can significantly enhance retrieval performance when compared against traditional and state-of-the-art, supervised and unsupervised data fusion approaches for microblog post search. Additional analyses showed that TimeRA is a robust and efficient data fusion method that outperforms state-of-the-art temporal retrieval algorithms.

After investigating data fusion for microblog search, we then turned to more generic issues of using data fusion in an ad hoc document retrieval setting. Recent state-of-the-art work tries to improve data fusion performance based on clustering: documents appearing in the lists to be fused are clustered and a document that appears low in a single list can be promoted if it is similar to other relevant documents in the cluster. While intuitive, such a fusion strategy can fall short in some cases. For instance, a non-relevant document should not be "promoted" even if it is in a cluster that contains a large number of relevant documents. What is worse, cluster-based data fusion may be a bottleneck in some applications, because of its computational costs. After investigating cluster-based data fusion method analytically and experimentally, we asked the following questions:

RQ 3 Can manifolds be used to improve data fusion performance for ad hoc search?

We have introduced a novel data fusion approach, ManX, which is based on manifold strategies to merge ranked lists that are retrieved in response to a given query. In ManX, manifolds of documents in the lists to be fused are constructed by utilizing inter-document similarities that are computed based on a multinomial distribution of topics specific to documents to be fused. ManX fully utilizes two prior assumptions underlying the cluster hypothesis, thereby enabling it to reward documents that are ranked low in only few lists but surrounding which there are many highly relevant documents in the same manifold. Furthermore, ManX takes top-k documents from fused lists merged with a standard fusion method X as anchor documents to make fusion process become faster.

We analyzed experimental results when fusing the lists submitted to the TREC-3 ad hoc, TREC-10 web and TREC-12 robust retrieval tracks, and demonstrated that our ManX method not only outperforms the standard fusion methods that it integrates and state-of-the-art data fusion method that leverages clustering strategies, but also fuses result lists more efficiently.

Data fusion methods can improve retrieval performance in terms of traditional relevance-oriented metrics like MAP and precision@k over the methods used to generate the individual result lists being fused. We also looked at the performance of data fusion methods in search result diversification and asked:

RQ 4 Can data fusion help search result diversification?

In this thesis we have adopted a new perspective on the search result diversification problem, based on data fusion. We proposed to use traditional unsupervised and state-of-the-art data fusion methods, CombSUM, CombMNZ, ClustFuseCombSUM, ClustFuseCombMNZ, CombSUMMMR and CombSUMPM-2 to diversify result lists. This led to the insight that fusion does aid diversification. We also proposed a fusion-based diversification method, DDF, which infers latent topics from ranked lists of documents produced by a standard fusion method, and combines this with a state-of-the-art result diversification model.

After a systematic analysis, we found that data fusion approaches outperform state-of-the-art search result diversification algorithms, with DDF invariably giving rise to the highest scores on all of the metrics that we have considered in this chapter. DDF was shown to behave well with different numbers of component lists. We also found that DDF is insensitive to the number of latent topics of a query, once a sufficiently large number was chosen, e.g., 10.

Finally, we zoomed in on studying the problem of personalized diversification of search results, with the goal of enhancing both diversification and personalization performance. In both search result diversification and personalized web search, an issued query is often viewed as an incomplete expression of a user's underlying need (Shen et al., 2005). Unlike search result diversification, where the system accepts and adapts its behavior to a situation of uncertainty, personalized web search strives to change this situation by enhancing the system's knowledge about users' information needs. Therefore, we asked the following research questions:

RQ 5 How to enhance both diversification and personalization performance at the same time in a supervised way?

In this thesis, we have adopted a different perspective on the problem, based on structured learning. We proposed to boost the diversity and matched to users' personal interests of search results by introducing two additional constraints into the standard structured learning framework. We also proposed a user-interest topic model to capture users' multinomial distribution of interest over topics and inferred per-document multinomial distributions over topics. Based on this a number of user interest features are extracted and the similarity between a user and a document can be effectively measured for our learning method.

We investigated the effects of our proposed learning framework for personalized diversification and found that supervised personalized diversification approaches outperforms state-of-the-art unsupervised personalization diversification, plain personalization and plain diversification algorithms. The two proposed constraints were shown to play a significant role in the supervised method. We also found that the user-interest topic model helps to improve performance. Our proposed learning method is able to return more subtopics and more diversified documents than state-of-the-art unsupervised personalized diversification, plain personalized web search, and plain search result diversification methods.

9.2 Future Work

The research presented in this thesis motivates a broad variety of future research projects, most of them aimed at improving over the results presented in the previous chapters, by adding new methods or optimizing existing ones. We do not list each of these smaller research directions, but focus on four major directions for future research in data fusion and search result diversification.

Data fusion. The effectiveness of retrieval methods often varies across queries (Harman and Buckley, 2004; Voorhees, 2005). Thus, the ability to automatically infer which queries are more difficult for a retrieval method than other queries can be very important. These observations have motivated a large body of work on predicting query performance (Markovits et al., 2012); that is, estimating the effectiveness of a search performed in response to a query in the absence of relevance judgements. Therefore, one important direction for following up on this work is to predict the performance of data fusion approaches. The goal is to predict the effectiveness of a document list that is produced by fusing a few lists that were retrieved from the same corpus or a different corpus by different retrieval systems in response to the query a user submitted. The motivation for the prediction of data fusion performance is at least twofold. First, fusion methods achieve effective retrieval performance on average when fusing lists retrieved by using different query and document representations or retrieval methods (Markovits et al., 2012). Furthermore, although fusion-based retrieval was shown to somewhat improve retrieval performance robustness across queries, we empirically showed that commonly used fusion methods still suffer from substantial performance variance. For instance, as shown in Section 4.3.4, although there are improvements for a majority of test queries, there is still a small number of queries suffer from poorer performance after data fusion.

Another important direction for data fusion is to utilize relevance feedback to boost the performance of data fusion. Using positive relevance feedback, such as document

clicks from users (Hofmann et al., 2013, 2014), has been shown in previous work in ad hoc retrieval to substantially improve retrieval effectiveness (Rabinovich et al., 2014; Ruthven and Lalmas, 2003). The idea of relevance feedback for data fusion is to involve the user in the fusion process so as to improve the quality of the final ranking of the documents. In particular, the user gives feedback on the relevance of documents in an initial set of results. Relevance feedback can go through one or more iterations by users. The process exploits the idea that it may be difficult to formulate a good query when you do not know the collection well, but it is easy to judge particular documents, and so it makes sense to engage in iterative query refinement in the data fusion process. In the scenario of fusing documents while involving user feedback, relevance feedback can also be effective in tracking a user's evolving information need, resulting in a final ranking of the documents that may be better at satisfying a user's information need than without adaptation over time.

Additionally, supervised learning for data fusion is also an important direction for future work beyond this thesis. The data fusion algorithms proposed in this thesis mainly belong to unsupervised learning approaches, in the sense that no training data is utilized. In fact, most previous work on data fusion also mainly focuses on unsupervised data fusion strategies. Methods like CombSUM, CombMNZ, Borda Count, median rank aggregation, genetic algorithm, fuzzy logic-based data fusion, Markov Chain-based rank aggregation are all unsupervised. Although there are several recent publications on data fusion via supervised learning strategies, like those proposed in (Liu et al., 2007) and (Hong and Si, 2012), it is still valuable to improve the performance of data fusion via other more effective supervised learning strategies. Utilizing supervised learning has a couple of advantages. For instance, we can leverage the use of information existing in labeled training data, can apply new optimization techniques to the problem of fusing documents, and it becomes easier to perform domain or user adaption.

Search result diversification. Although a number of search result diversification techniques have been proposed, there are several challenges that need to be dealt with. One direction of future work on search result diversification is to estimate the number of aspects underlying a query and the weights of the aspects. The performance of previous search result diversification algorithms, especially for the explicit approaches, mainly depends on the availability of the number of aspects underlying an ambiguous query and the corresponding weights (Dang and Croft, 2012, 2013). However, estimating the aspects and the weights is a bottleneck in current search result diversification algorithms (Santos et al., 2010a). The behavior of users using the search engines suggests some ways to tackle the challenges. For instance, after submitting a query to the system, some users may not be satisfied with the search results, and then they would iteratively reformulate their queries until the documents returned by the system satisfy their information need. During these iterations, the user may click on some documents. Hence, exploiting and mining users' query representations, the click data and the search logs could result in valuable ways to estimate the aspects and the corresponding weights.

Another direction for future work is to personalize search result diversification in social media. Two main components, i.e., personalized web search and search result diversification, play important roles in tackling the problem of personalized search result diversification. Although there is a large body of work on either personalized web search

or search result diversification separately, only three previous works have studied the problem of combining both personalization and diversification. Radlinski and Dumais (2006) analyzed a large sample of individual users' query logs from a web search engine such that the individual users' query reformulations can be obtained. Then they personalized web search by re-ranking the top results using query reformulations to introduce diversity into those results. Vallet and Castells (2012) investigated the introduction of the user as an explicit variable in existing search result diversification models. Liang et al. (2014b) set up a structured learning framework for conducting supervised personalized diversification. All of the proposed personalized diversification approaches suppose that users are isolated, and ignore social relationship among users. However, in the context of social media, users usually interact with each other. Hence, in order to further improve the performance of personalized diversification, social network data may be a valuable source of information.

Finally, other directions of future work include, for instance, to estimate the scores of diversification evaluation metrics. In recent years, a number of evaluation metrics have been proposed and many of them have been adopted by the diversify task at the Web 2009-2013 tracks in TREC to evaluate the submitted runs. As explained in Chapter 3, these official evaluation metrics include, e.g., subtopic recall at k (S-Recall@k), normalized discounted cumulative gain at k (α-nDCG@k), intent-aware expected reciprocal rank at retrieval depth k (ERR-IA@k), intent-aware precision at k (Prec-IA@k), and intent-aware MAP at k (MAP-IA@k). All of the proposed diversification evaluation metrics make a strong assumption that the relevance judgements are complete, i.e., for each query, all aspects of the query are easily identified, and all documents in the collection relevant to these aspects are easily identified as well. However, most diversification algorithms are applied in the web search applications. In these cases, the collection is dynamic. For instance, an increasing number of unjudged or unseen documents and aspects are added into it, resulting in the fact that obtaining complete relevance judgments is infeasible for each aspect of the query because of the need of extensive human effort (Yilmaz et al., 2008). Specifically, we would like to infer the efficiency of the diversification metrics and the corresponding evaluation scores given a search results.

Bibliography

S. Abbar, S. Amer-Yahia, P. Indyk, and S. Mahabadi. Real-time recommendation of diverse related articles. In *WWW*, pages 1–12, 2013. (Cited on pages 22 and 105.)

R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM*, pages 5–14, 2009. (Cited on pages 2, 22, 23, 33, 101, 109, and 131.)

N. Ahmad and M. M. S. Beg. Fuzzy logic based rank aggregation methods for the world wide web. In *Proceedings of the International Conference on Artificial Intelligence in Engineering and Technology*, pages 363–368, 2002. (Cited on page 17.)

E. Aktolga and J. Allan. Sentiment diversification with different biases. In *SIGIR*, pages 593–600, 2013. (Cited on page 109.)

G. Amati, G. Amodeo, M. Bianchi, A. Celi, C. D. Nicola, M. Flammini, C. Gaibisso, G. Gambosi, and G. Marcone. FUB, IASI-CNR, UNIVAQ at TREC 2011 microblog track. In *Proceedings of the Text REtrieval Conference*, 2011. (Cited on pages 19 and 55.)

N. Asadi, D. Metzler, T. Elsayed, and J. Lin. Pseudo test collections for learning web search ranking functions. In *SIGIR'11*, pages 1073–1081, 2011. (Cited on page 59.)

J. A. Aslam and M. Montague. Models for metasearch. In *SIGIR'01*, pages 276–284, 2001. (Cited on pages 16 and 17.)

P. Atrey, M. Hossain, A. E. Saddik, and M. Kankanhalli. Multimodal fusion for multimedia analysis: a survey. *Multimedia Syst.*, 16(6):345–379, 2010. (Cited on page 61.)

K. Bache, D. Newman, and P. Smyth. Text-based measures of document diversity. In *KDD*, pages 23–31, 2013. (Cited on page 22.)

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology behind Search (2nd Edition) (ACM Press Books)*. Addison-Wesley Professional, 2 edition, Feb. 2011. ISBN 0321416910. (Cited on pages 11, 28, and 31.)

K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 43–50, 2006. (Cited on page 1.)

A. Bandyopadhyay, M. Mitra, and P. Majumder. Query expansion for microblog retrieval. In *TREC'11*, 2011. (Cited on page 19.)

M. M. S. Beg. Parallel rank aggregation for the world wide web. In *Proceedings of the Intelligent Sensing and Information Processing*, pages 385–390, 2004. (Cited on page 16.)

S. M. Beitzel, E. C. Jensen, A. Chowdhury, O. Frieder, D. A. Grossman, and N. Goharian. Disproving the fusion hypothesis: An analysis of data fusion via effective information retrieval strategies. In *SAC'03*, pages 823–827, 2003. (Cited on pages 2 and 15.)

P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short- and long-term behavior on search personalization. In *SIGIR*, pages 185–194, 2012. (Cited on pages 3 and 23.)

R. Berendsen, M. Tsagkias, W. Weerkamp, and M. de Rijke. Pseudo test collections for training and tuning microblog rankers. In *SIGIR'13*, pages 53–62, 2013. (Cited on page 59.)

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003. (Cited on pages 24, 25, 26, 122, and 129.)

J. Boyd-Graber and D. M. Blei. Syntactic topic models. In *NIPS*, pages 185–192, 2008. (Cited on page 127.)

J. Boyd-Graber and D. M. Blei. Multilingual topic models for unaligned text. In *UAI*, pages 75–82, 2009. (Cited on page 127.)

E. Bruno and S. Marchand-Maillet. Multiview clustering: a late fusion approach using latent models. In *SIGIR'09*, pages 736–737, 2009. (Cited on pages 16 and 38.)

F. Cai, S. Liang, and M. de Rijke. Personalized document re-ranking based on bayesian probabilistic matrix factorization. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 835–838, 2014a. (Cited on page 8.)

F. Cai, S. Liang, and M. de Rijke. Time-sensitive personalized query auto-completion. In *Proceedings of the 23rd ACM International Conference on Information & Knowledge Management*, CIKM '14, page 10, 2014b. (Cited on page 8.)

L. Cao, R. Ji, W. Liu, H. Yao, and Q. Tian. Weakly supervised codebook learning by iterative label propagation with graph quantization. *Signal Processing*, 93(8):2274 – 2283, 2013. (Cited on page 83.)

P. Cao, J. Gao, Y. Yu, S. Liu, Y. Liu, and X. Cheng. ICTNET at microblog track TREC 2011. In *Proceedings of the Text REtrieval Conference*, 2011. (Cited on pages 19 and 55.)

J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and

producing summaries. In *SIGIR*, pages 335–336, 1998. (Cited on pages 22, 24, 102, and 110.)

C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. Predicate rewriting for translating boolean queries in a heterogeneous information system. *ACM Transactions on Information Systems (TOIS)*, 17(1):1–39, 1999. (Cited on page 12.)

Y. Chang, A. Dong, P. Kolari, R. Zhang, Y. Inagaki, F. Diaz, H. Zha, and Y. Liu. Improving recency ranking using twitter data. *ACM Trans. Intell. Syst. Technol.*, 4(1):4:1–4:24, Feb. 2013. (Cited on page 20.)

H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In *SIGIR*, pages 429–436, 2006. (Cited on page 22.)

W. Chen, C. Chen, L. jun Zhang, C. Wang, and J. jun Bu. Online detection of bursty events and their evolution in news streams. *Journal of Zhejiang University*, 11:340–355, 2010. (Cited on page 36.)

J. Choi and W. B. Croft. Temporal models for microblogs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2491–2494, New York, NY, USA, 2012. ACM. (Cited on page 20.)

J. Choi, W. B. Croft, and J. Y. Kim. Quality models for microblog retrieval. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 1834–1838, New York, NY, USA, 2012. ACM. (Cited on page 19.)

F. R. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997. (Cited on page 87.)

C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 659–666. ACM, 2008a. (Cited on page 32.)

C. L. A. Clarke and N. Craswell. Overview of the TREC 2011 web track. In *TREC*, pages 1–9, 2011. (Cited on pages 13, 22, 23, 30, 32, 109, and 110.)

C. L. A. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR*, pages 659–666, 2008b. (Cited on pages 33, 109, and 123.)

C. L. A. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2009 web track. In *TREC*, pages 1–9, 2009. (Cited on pages 30, 32, 109, and 110.)

C. L. A. Clarke, N. Craswell, I. Soboroff, and G. V. Cormack. Overview of the TREC 2010 web track. In *TREC*, pages 1–9, 2010. (Cited on pages 30, 32, 109, and 110.)

C. L. A. Clarke, N. Craswell, and E. M. Voorhees. Overview of the TREC 2012 web track. In *TREC*, pages 1–8, 2012. (Cited on pages 22, 23, 30, 32, 109, 110, 126, 131, and 132.)

F. Crestani and I. Markov. Distributed information retrieval and applications. In *ECIR*, pages 865–868. Springer Berlin Heidelberg, 2013. (Cited on pages 16 and 59.)

W. B. Croft. *Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval*. Kluwer, 2000. (Cited on pages 15, 16, and 17.)

J. S. Culpepper, M. Petri, and F. Scholer. Efficient in-memory top-k document retrieval. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 225–234. ACM, 2012. (Cited on page 87.)

W. Dakka, L. Gravano, and P. Ipeirotis. Answering general time-sensitive queries. *IEEE Trans. Knowledge and Data Engin.*, 24(2):220–235, 2012. (Cited on pages 20, 45, 56, 70, and 77.)

V. Dang and W. B. Croft. Diversity by proportionality: An election-based approach to search result diversification. In *SIGIR*, pages 65–74, 2012. (Cited on pages 22, 23, 24, 26, 102, 103, 109, 110, and 147.)

V. Dang and W. B. Croft. Term level search result diversification. In *SIGIR*, pages 603–612, 2013. (Cited on pages 22, 24, 109, and 147.)

P. Das-Gupta and J. Katzer. A study of the overlap among document representations. In *SIGIR'83*, pages 106–114, 1983. (Cited on page 4.)

F. Diaz. Regularizing ad hoc retrieval scores. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 672–679. ACM, 2005. (Cited on page 21.)

X. L. Dong and D. Srivastava. Compact explanation of data fusion decisions. In *WWW'13*, pages 379–390, 2013. (Cited on pages 2, 3, and 15.)

P. Donmez, K. M. Svore, and C. J. C. Burges. On the local optimality of lambdarank, 2008. Microsoft Research Technical Report. (Cited on page 62.)

Y. Duan, L. Jiang, T. Qin, M. Zhou, and H. Shum. An empirical study on learning to rank tweets. In *COLING*, pages 295–303, 2010. (Cited on page 19.)

C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *WWW'01*, pages 613–622, 2001. (Cited on pages 16 and 17.)

M. Efron. Hashtag retrieval in a microblogging environment. In *SIGIR'10*, pages 787–788, 2010. (Cited on

page 19.)
M. Efron. Information search and retrieval in microblogs. *J. Am. Soc. for Inform. Sci. and Techn.*, 62(6): 996–1008, 2011. (Cited on pages 16 and 68.)
R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 301–312, New York, NY, USA, 2003. ACM. ISBN 1-58113-634-X. (Cited on page 16.)
M. Farah and D. Vanderpooten. An outranking approach for rank aggregation in information retrieval. In *SIGIR'07*, pages 591–598, 2007. (Cited on page 17.)
E. A. Fox and J. A. Shaw. Combination of multiple searches. In *TREC-2*, 1994. (Cited on pages 1, 3, 4, 5, 15, 16, 17, 38, 63, and 68.)
D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, Dec. 1992. (Cited on pages 20 and 63.)
W. S. Gosset. The application of the law of error to the work of the brewery. *Guinness Internal Note*, 1904. (Cited on pages 33 and 34.)
A. Griffiths, H. C. Luckhurst, and P. Willett. Using interdocument similarity information in document retrieval systems. *Journal of the American Society for Information Science*, 37(1):3–11, 1986. (Cited on page 4.)
A. Griffiths, H. C. Luckhurst, and P. Willett. Adaptive manifold learning. *IEEE trans. on PAMI*, 34(2):253–265, 2012. (Cited on page 21.)
T. L. Griffiths and M. Steyvers. Finding scientific topics. *PNAS*, 101:5228–5235, 2004. (Cited on pages 24, 26, and 106.)
Z. Han, X. Li, M. Yang, H. Qi, S. Li, and T. Zhao. HIT at TREC 2012 microblog track. In *TREC '12 Working Notes*, 2012. (Cited on pages 19 and 29.)
D. Harman. Overview of the third text retrieval conference (TREC-3). In *TREC'94*, 1994. (Cited on pages 28 and 90.)
D. Harman and C. Buckley. The NRRC reliable information access (RIA) workshop. In *Proceedings of SIGIR*, pages 528–529, 2004. (Cited on page 146.)
D. Hawking and N. Craswell. Overview of the TREC-2001 web track. In *Proceedings of the TREC-10*, pages 1–8, 2002. (Cited on pages 28 and 90.)
C. He, D. Hong, and L. Si. A weighted curve fitting method for result merging in federated search. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 1177–1178, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0757-4. (Cited on page 16.)
D. He and D. Wu. Toward a robust data fusion for document retrieval. In *IEEE NLP-KE'08*, pages 1–8, 2008. (Cited on pages 1, 4, and 15.)
J. He, V. Hollink, and A. de Vries. Combining implicit and explicit topic representations for result diversification. In *SIGIR*, pages 851–860, 2012. (Cited on page 105.)
K. Hofmann, S. Whiteson, and M. D. Rijke. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Trans. Inf. Syst.*, 31(4):17:1–17:43, Nov. 2013. (Cited on pages 87 and 147.)
K. Hofmann, A. Schuth, A. Bellogin, and M. de Rijke. Effects of position bias on click-based recommender evaluation. In *36th European Conference on Information Retrieval (ECIR'14)*, pages 624–630, 2014. (Cited on page 147.)
T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999. (Cited on page 24.)
T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, Jan. 2004. (Cited on page 79.)
D. Hong and L. Si. Mixture model with multiple centralized retrieval algorithms for result merging in federated search. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 821–830, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1472-5. (Cited on pages 16 and 147.)
D. Hong and L. Si. Search result diversification in resource selection for federated search. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 613–622, New York, NY, USA, 2013. ACM. (Cited on page 16.)
A. Hoonlor, B. K. Szymanski, M. J. Zaki, and V. Chaoji. Document clustering with bursty. *Computing and Informatics*, 31:1533–1555, 2012. (Cited on page 36.)
C. Horn, O. Pimas, M. Granitzer, and E. Lex. Realtime ad hoc search in Twitter: Know-center at TREC microblog track 2011. In *Proceedings of the Text REtrieval Conference*, 2011. (Cited on pages 19 and 55.)
L. B. Jabeur, F. Damak, L. Tamine, K. Pinel-Sauvagnat, G. Cabanac, and M. Boughanem. IRIT at TREC microblog 2012: Adhoc task. In *TREC'12*, 2012. (Cited on page 19.)
S. Jameel and W. Lam. An unsupervised topic segmentation model incorporating word order. In *SIGIR*, pages

203–212, 2013. (Cited on page 127.)
K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4): 422–446, 2002. (Cited on pages 32 and 132.)
O. Jin, N. N. Liu, K. Zhao, Y. Yu, and Q. Yang. Transferring topical knowledge from auxiliary long texts for short text clustering. In *CIKM*, pages 775–784, 2011. (Cited on page 24.)
T. Joyce and R. Needham. The thesaurus approach to information retrieval. *American Documentation*, 9(3): 192–197, 1958. (Cited on pages 1 and 11.)
J. Karas and I. Savage. Publications of frank wilcoxon. *Biometrics*, 23:1–10, 1967. (Cited on page 33.)
S. Khalaman and O. Kurland. Utilizing inter-document similarities in federated search. In *SIGIR'12*, pages 1169–1170, 2012. (Cited on pages 16, 17, 42, and 43.)
S. Khuller, A. Moss, and J. S. Naor. The budgeted maximum coverage problem. *Inf. Proc. Lett.*, 70(1):39–45, 1999. (Cited on page 125.)
Y. Kim, R. Yeniterzi, and J. Callan. Overcoming vocabulary limitations in twitter microblogs. In *TREC'12*, 2012. (Cited on page 19.)
J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, Sept. 1999. (Cited on page 86.)
A. Klementiev, D. Roth, and K. Small. Unsupervised rank aggregation with distance-based models. In *ICML'08*, pages 472–479, 2008. (Cited on page 17.)
A. K. Kozorovitsky and O. Kurland. Cluster-based fusion of retrieved lists. In *SIGIR'11*, pages 893–902, 2011. (Cited on pages 1, 4, 15, 16, 17, 18, 38, 68, 70, 81, 82, 83, and 110.)
T. Kurashima, T. Iwata, T. Hoshide, N. Takaya, and K. Fujimura. Geo topic model: joint modeling of user's activity area and interests for location recommendation. In *WSDM*, pages 375–384, 2013. (Cited on page 25.)
O. Kurland and L. Lee. Corpus structure, language models, and ad hoc information retrieval. In *SIGIR'04*, pages 194–201, 2004. (Cited on page 42.)
M. Kurucz, A. A. Benczúr, and K. Csalogány. Methods for large scale svd with missing values. In *KDD Cup and Workshop*, volume 12, pages 31–38, 2007. (Cited on pages 20, 63, and 66.)
J. D. Lafferty and D. M. Blei. Correlated topic models. In *Advances in neural information processing systems*, pages 147–154, 2005. (Cited on page 25.)
T. Lappas, B. Arai, M. Platakis, D. Kotsakos, and D. Gunopulos. On burstiness-aware search for document sequences. In *SIGKDD'09*, pages 477–486, 2009. (Cited on pages 36, 41, 45, and 57.)
G. Lee and J. S. et al. Siteq: Engineering high performance qa system using lexico-semantic pattern matching and shallow nlp. In *TREC*, 2011. (Cited on page 62.)
J. H. Lee. Combining multiple evidence from different properties of weighting schemes. In *SIGIR'95*, pages 180–188, 1995. (Cited on pages 5, 16, 17, 37, 38, and 110.)
J. H. Lee. Analyses of multiple evidence combination. In *SIGIR'97*, pages 267–276, 1997. (Cited on page 5.)
F. Li, M. Huang, and X. Zhu. Sentiment analysis with global topics and local dependency. In *AAAI*, pages 1371–1376, 2010. (Cited on page 25.)
W. Li and A. McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *ICML*, pages 577–584. ACM, 2006. (Cited on page 25.)
X. Li and W. B. Croft. Time-based language models. In *CIKM*, pages 469–475, 2003. (Cited on pages 45, 56, 70, and 77.)
S. Liang and M. de Rijke. Finding knowledgeable groups in enterprise corpora. In *SIGIR'13*, 2013. (Cited on pages 9 and 59.)
S. Liang and M. de Rijke. Formal language models for finding groups of experts. *Some journal*, Submitted. (Cited on page 8.)
S. Liang and M. de Rijke. Burst-aware data fusion for microblog search. *Information Processing & Management*, To appear. (Cited on page 8.)
S. Liang, M. de Rijke, and M. Tsagkias. Late data fusion for microblog search. In *Proceedings of the 35th European Conference on Information Retrieval*, pages 743–746, 2013. (Cited on pages 9 and 15.)
S. Liang, Z. Ren, and M. de Rijke. Fusion helps diversification. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 303–312, 2014a. (Cited on pages 8 and 23.)
S. Liang, Z. Ren, and M. de Rijke. Personalized search result diversification via structured learning. In *Proceedings of the 20th International ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 751–760, 2014b. (Cited on pages 8 and 148.)
S. Liang, Z. Ren, and M. de Rijke. The impact of semantic document expansion on cluster-based fusion for microblog search. In *ECIR'14*, pages 493–499, 2014c. (Cited on page 9.)
S. Liang, Z. Ren, W. Weerkamp, E. Meij, and M. de Rijke. Time-aware rank aggregation for microblog search.

In *Proceedings of the 23rd ACM International Conference on Information & Knowledge Management*, CIKM '14, page 10, 2014d. (Cited on page 8.)

S. Liang, I. Markov, Z. Ren, and M. de Rijke. Efficient manifold-based fusion of ranked lists. *Some journal*, Submitted. (Cited on page 8.)

N. Limsopatham, R. McCreadie, and M.-D. Albakour. University of Glasgow at TREC 2012: Experiments with Terrier in medical records, microblog, and web tracks. In *TREC*, 2012. (Cited on page 111.)

J. Lin, C. Macdonald, I. Ounis, and I. Soboroff. Overview of the TREC 2011 Microblog track. In *TREC 2011*. NIST, 2012. (Cited on pages 3, 19, 29, 35, and 62.)

C. Liu, N. J. Belkin, and M. J. Cole. Personalization of search results using interaction behaviors in search sessions. In *SIGIR*, pages 205–214, 2012a. (Cited on page 23.)

J. S. Liu. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *J. Am. Stat. Assoc.*, 89(427):958–966, 1994. (Cited on pages 105, 106, 126, and 127.)

T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3): 225–331, 2009. (Cited on pages 11 and 13.)

W. Liu, J. He, and S.-F. Chang. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 679–686, 2010. (Cited on pages 83 and 87.)

W. Liu, J. Wang, and S.-F. Chang. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, 100(9):2624–2638, 2012b. (Cited on page 83.)

X. Liu and W. B. Croft. Evaluating text representations for retrieval of the best group of documents. In *ECIR'08*, pages 454–462, 2008. (Cited on page 43.)

Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, and H. Li. Supervised rank aggregation. In *WWW'07*, pages 481–489, 2007. (Cited on pages 16 and 147.)

Z. Luo, M. Osborne, S. Petrovic, and T. Wang. Improving twitter retrieval by exploiting structural information. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 648–654, 2012. (Cited on page 19.)

H. Ma, D. Zhou, and C. Liu. Recommender systems with social regularization. In *WSDM'11*, pages 287–296, 2011a. (Cited on pages 20, 63, and 66.)

H. Ma, T. C. Zhou, M. R. Lyu, and I. King. Improving recommender systems by incorporating social contextual information. *ACM Trans. Inf. Syst.*, 29(2):9:1–9:23, Apr. 2011b. (Cited on page 70.)

C. Macdonald, I. Ounis, J. Lin, A. Choudhury, and I. Soboroff. 2011 track guidelines, 2011. `http://trec.nist.gov`. (Cited on pages 28, 29, and 62.)

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715. (Cited on pages 1, 11, and 31.)

I. Markov and F. Crestani. Theoretical, qualitative and quantitative analyses of small-document approaches to resource selection. *ACM Trans. Inf. Syst.*, 32(2):9:1–9:37, 2014. (Cited on pages 16 and 17.)

I. Markov, A. Arampatzis, and F. Crestani. Unsupervised linear score normalization revisited. In *SIGIR*, pages 1161–1162. ACM, 2012. (Cited on page 17.)

I. Markov, A. Arampatzis, and F. Crestani. On cori result merging. In *ECIR*, pages 752–755. Springer, 2013a. (Cited on page 17.)

I. Markov, L. Azzopardi, and F. Crestani. Reducing the uncertainty in resource selection. In *ECIR*, pages 507–519. Springer Berlin Heidelberg, 2013b. (Cited on page 16.)

G. Markovits, A. Shtok, and O. Kurland. Predicting query performance for fusion-based retrieval. In *CIKM'12*, pages 813–822, 2012. (Cited on pages 42 and 146.)

M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)*, 7(3):216–244, 1960. (Cited on pages 1, 11, and 12.)

K. Massoudi, M. Tsagkias, M. de Rijke, and W. Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. In *ECIR '11*, pages 362–367, 2011. (Cited on pages 19, 20, 45, 56, 70, and 77.)

M. Mathioudakis, N. Bansal, and N. Koudas. Identifying, attributing and describing spatial bursts. In *VLDB'10*, pages 1091–1102, 2010. (Cited on pages 36 and 41.)

E. Meij, W. Weerkamp, and M. de Rijke. Adding semantics to microblog posts. In *WSDM '12*, pages 563–572. ACM, 2012. (Cited on page 99.)

D. Metzler and C. Cai. USC/ISI at TREC 2011: microblog track. In *Proceedings of the Text REtrieval Conference*, 2011. (Cited on pages 19 and 55.)

D. Metzler, C. Cai, and E. Hovy. Structured event retrieval over microblog archives. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 646–655, 2012. (Cited on pages 39 and 55.)

T. Miyanishi, K. Seki, and K. Uehara. Combing recency and topic-dependent temporal variation for microblog search. In *ECIR'13*, pages 331–343, 2013a. (Cited on page 19.)

T. Miyanishi, K. Seki, and K. Uehara. Improving pseudo-relevance feedback via tweet selection. In *CIKM*, pages 439–448, 2013b. (Cited on pages 20, 45, 56, 70, and 77.)

M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In *CIKM'02*, pages 538–548, 2002. (Cited on pages 2, 15, and 17.)

C. Mooers. The next twenty years in information retrieval. *American Documentation*, 11(3):229–236, 1960. (Cited on page 11.)

N. Naveed, T. Gottron, J. Kunegis, and A. Che Alhadi. Searching microblogs: coping with sparsity and document quality. In *CIKM'11*, pages 183–188. ACM, 2011. (Cited on page 19.)

B. O'Connor, M. Krieger, and D. Ahn. TweetMotif: Exploratory search and topic summarization for Twitter. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 384–385, 2010. (Cited on page 19.)

M.-H. Peetz, E. Meij, M. de Rijke, and W. Weerkamp. Adaptive temporal query modeling. In *ECIR '12*, pages 455–458, 2012. (Cited on pages 36 and 41.)

J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 275–281, 1998. (Cited on pages 11 and 12.)

T. Qin, X. Geng, and T.-Y. Liu. A new probabilistic model for rank aggregation. In *NIPS'10*, pages 1948–1956, 2010. (Cited on page 16.)

E. Rabinovich, O. Rom, and O. Kurland. Utilizing relevance feedback in fusion-based retrieval. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 313–322, 2014. (Cited on page 147.)

F. Radlinski and S. Dumais. Improving personalized web search using result diversification. In *SIGIR*, pages 691–692. ACM, 2006. (Cited on pages 23, 121, 122, 131, and 148.)

Z. Ren, S. Liang, E. Meij, and M. de Rijke. Personalized time-aware tweets summarization. In *SIGIR'13*, pages 513–522, 2013. (Cited on pages 9, 25, and 106.)

Z. Ren, M.-H. Peetz, S. Liang, W. van Dolen, and M. de Rijke. Hierarchical multi-label classification of social text streams. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 213–222, 2014. (Cited on page 9.)

S. E. Robertson. The probability ranking principle in ir. *Journal of documentation*, 33(4):294–304, 1977. (Cited on pages 2 and 12.)

S. E. Robertson and D. A. Hull. The TREC-9 filtering track final report. In *TREC*, pages 25–40, 2000. (Cited on pages 129 and 131.)

S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976. (Cited on page 12.)

S. E. Robertson, C. Van Rijsbergen, and M. F. Porter. Probabilistic models of indexing and searching. In *Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*, pages 35–56. Butterworth & Co., 1980. (Cited on page 12.)

M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *UAI*, pages 487–494, 2004. (Cited on page 25.)

I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18(2):95–145, June 2003. (Cited on page 147.)

W. L. Ruzzo and M. Tompa. A linear time algorithm for finding all maximal scoring subsequences. In *Int. Conf. Intelligent Systems for Molecular Biology*, pages 234–241, 1999. (Cited on pages 41 and 69.)

T. Sakai, Z. Dou, and C. L. A. Clarke. The impact of intent selection on diversified search result. In *SIGIR*, pages 921–924, 2013. (Cited on page 109.)

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *ICML*, pages 880–887, 2008a. (Cited on pages 20, 37, 63, and 66.)

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2008b. (Cited on pages 20, 63, and 66.)

G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM (JACM)*, 15(1):8–36, 1968. (Cited on pages 1, 11, and 12.)

R. L. Santos, C. Macdonald, and I. Ounis. Exploiting query reformulations for web search result diversification. In *WWW*, pages 881–890, 2010a. (Cited on pages 22, 23, 26, 105, 110, 131, and 147.)

R. L. Santos, C. Macdonald, and I. Ounis. Intent-aware search result diversification. In *SIGIR*, pages 595–604, 2011. (Cited on pages 101 and 109.)

R. L. T. Santos, J. Peng, C. Macdonald, and I. Ounis. Explicit search result diversification through sub-queries.

In *ECIR*, pages 87–99, 2010b. (Cited on page 22.)
J. Seo and W. B. Croft. Geometric representations for multiple documents. In *SIGIR'10*, pages 251–258, 2010. (Cited on page 43.)
J. A. Shaw and E. A. Fox. Combination of multiple searches. In *TREC 1992*, pages 243–252. NIST, 1993. (Cited on pages 1, 4, 15, 16, 35, and 37.)
D. Sheldon, M. Shokouhi, M. Szummer, and N. Craswell. LambdaMerge: merging the results of query reformulations. In *WSDM '11*, pages 795–804, 2011. (Cited on pages 1, 15, 16, 37, 38, 39, 45, 62, 68, and 70.)
X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *CIKM*, pages 824–831. ACM, 2005. (Cited on pages 5, 121, and 145.)
Y. Shi, X. Zhao, J. Wang, M. Larson, and A. Hanjalic. Adaptive diversification of recommendation results via latent factor portfolio. In *SIGIR*, pages 175–184. ACM, 2012. (Cited on page 121.)
M. Shokouhi and L. Si. Federated search. *Found. Trends Inf. Retr.*, 5(1):1–102, Jan. 2011. (Cited on pages 2, 15, 16, and 59.)
L. Si, J. Callan, S. Cetintas, and H. Yuan. An effective and efficient results merging strategy for multilingual information retrieval in federated search environments. *Information Retrieval*, 11(1):1–24, 2008. (Cited on pages 16 and 59.)
M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 623–632. ACM, 2007. (Cited on page 33.)
I. Soboroff, I. Ounis, C. Macdonald, and J. Lin. Overview of the TREC-2012 Microblog track. In *TREC 2012*. NIST, 2012. (Cited on pages 13, 29, and 35.)
I. Szpektor, Y. Maarek, and D. Pelleg. When relevance is not enough: promoting diversity and freshness in personalized question recommendation. In *WWW '13*, pages 1249–1260, 2013. (Cited on page 23.)
W. P. Thurston and J. W. Milnor. *The geometry and topology of three-manifolds*. Princeton University Princeton, 1979. (Cited on pages 2, 21, and 81.)
M. Tsagkias, M. de Rijke, and W. Weerkamp. Linking online news and social media. In *WSDM '11*, pages 565–574, 2011. (Cited on pages 16 and 17.)
M.-F. Tsai, Y.-T. Wang, and H.-H. Chen. A study of learning a merge model for multilingual information retrieval. In *SIGIR'08*, pages 195–202, 2008. (Cited on pages 1, 4, 15, 16, 62, and 68.)
I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005. (Cited on pages 26, 122, 124, and 125.)
D. Vallet and P. Castells. Personalized diversification of search results. In *SIGIR'12*, pages 841–850, 2012. (Cited on pages 3, 5, 23, 30, 121, 122, 129, 131, 132, and 148.)
D. Vallet, I. Cantador, and J. M. Jose. Personalizing web search with folksonomy-based user and document profiles. In *ECIR*, pages 420–431. Springer, 2010. (Cited on pages 3, 23, and 131.)
M. van Erp and L. Schomaker. Variants of the borda count method for combining ranked classifier hypotheses. In *Proceedings of the 7th International Workshop on Frontiers in handwriting Recognition*, pages 443–452, 2000. (Cited on page 16.)
S. Vargas, P. Castells, and D. Vallet. Explicit relevance models in intent-oriented information retrieval diversification. In *SIGIR*, pages 75–84, 2012. (Cited on pages 22 and 26.)
M. Vlachos, C. Meek, and Z. Vagena. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD'04*, pages 131–142, 2004. (Cited on page 36.)
E. M. Voorhees. Overview of the TREC 2005 robust retrieval track. In *Proceedings of the TREC-14*, pages 1–9, 2005. (Cited on pages 28, 90, and 146.)
H. Wang, X. He, M.-W. Chang, Y. Song, R. W. White, and W. Chu. Personalized ranking model adaptation for web search. In *SIGIR*, pages 323–332, 2013. (Cited on page 23.)
R. Wang, S. Shan, X. Chen, and W. Gao. Manifold-manifold distance with application to face recognition based on image set. In *CVPR'08*, pages 1–8. IEEE, 2008. (Cited on pages 21 and 82.)
X. Wang and A. McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *KDD'06*, pages 424–433, 2006. (Cited on pages 25, 106, and 127.)
W. Weerkamp and M. de Rijke. Credibility-inspired ranking for blog post retrieval. *Information Retrieval Journal*, 15(3–4):243–277, 2012. (Cited on page 62.)
B. Wei, S. Zhang, R. Li, and B. Wang. A time-aware language model for microblog retrieval. In *TREC'12*, 2012. (Cited on page 19.)
X. Wei and W. B. Croft. Lda-based document models for ad-hoc retrieval. In *SIGIR*, pages 178–185, 2006. (Cited on pages 25 and 106.)
X. Wei, J. Sun, and X. Wang. Dynamic mixture models for multiple time-series. In *IJCAI*, pages 2909–2914,

2007. (Cited on page 25.)

Z. Wei, W. Gao, L. Zhou, B. Li, and K.-F. Wong. Exploring tweets normalization and query time sensitivity for twitter search. In *Proceedings of the Text REtrieval Conference*, 2011. (Cited on pages 19 and 55.)

F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83, 1945. (Cited on page 33.)

S. Wu. *Data fusion in information retrieval*, volume 13 of *Adaptation, Learning and Optimization*. Springer, 2012. (Cited on pages 2, 3, 4, 5, 15, 16, 17, 38, 63, 68, and 101.)

Z. Xu, Y. Zhang, Y. Wu, and Q. Yang. Modeling user posting behavior on social media. In *SIGIR*, pages 545–554, 2012. (Cited on pages 25 and 106.)

J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *WSDM '11*, pages 177–186. ACM, 2011. (Cited on pages 3 and 35.)

E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating ap and ndcg. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 603–610. ACM, 2008. (Cited on page 148.)

Y. Yue and T. Joachims. Predicting diverse subsets using structural svms. In *ICML*, pages 1224–1231. ACM, 2008. (Cited on pages 26, 122, 128, 129, and 131.)

Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR*, pages 271–278. ACM, 2007. (Cited on pages 26, 122, and 131.)

C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, pages 334–342, 2001. (Cited on page 129.)

C. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *SIGIR*, pages 10–17, 2003. (Cited on pages 22 and 33.)

S.-X. Zhang and M. Gales. Structured SVMs for automatic speech recognition. *IEEE Trans. on Audio, Speech, and Lang. Proc.*, 21(3):544–555, 2013. (Cited on page 26.)

X. Zhang, K. Hui, B. He, and T. Luo. GUCAS at TREC-2011 microblog track. In *TREC'11*, 2011. (Cited on page 19.)

D. Zhao and M. B. Rosson. How and why people twitter: the role that micro-blogging plays in informal communication at work. In *GROUP '09*, pages 243–252, 2009. (Cited on pages 2, 3, 17, and 35.)

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS 16*, pages 321–328. MIT Press, 2004. (Cited on pages 21, 22, 83, and 84.)

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML'03*, pages 912–919, 2003. (Cited on page 21.)

Y. Zhu, Y. Lan, J. Guo, X. Cheng, and S. Niu. Learning for search result diversification. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 293–302, 2014. (Cited on page 22.)

Summary

Data fusion and search result diversification are two critical research topics in information retrieval. The first main topic on which this thesis focuses is data fusion. Data fusion approaches combine search result lists in order to produce a new and hopefully better ranking. The result lists in data fusion can be produced by a wide range of ranking approaches, based, for instance, on different query or document representations. Many effective state-of-the-art data fusion approaches are based on the assumption that only documents that are highly ranked in many of the result lists are likely to be relevant. As a consequence, a relevant document will be ranked low in the final fused list if it appears only in a single list and is ranked low in this list. However, when considering the application of data fusion in microblog environments, this assumption may not lead to effective retrieval. We propose two novel data fusion models for microblog search that not only utilize information traditionally used when merging ranked lists, such as ranks, but also exploit temporal information, i.e., the publication time of microblog posts. In addition, previous work on data fusion often assumes, either implicitly or explicitly, that the rank score of a document is set to zero if the document does not appear in the result list. We challenge this assumption. We apply a latent factor model to predict the rank scores of such documents. Our intuition is that documents that are similar to a document that occurs in an input list, should get similar rank scores. Furthermore, previous work on data fusion has mainly focused on fusion based on retrieval status values or ranks of documents in the result lists without fully analyzing inter-document similarities. We propose a fusion method based on manifolds. The method constructs manifolds for documents based on their similarities, let low ranked documents be rewarded to be relevant by high ranked documents in the same manifolds, and utilize the top-k documents as anchors to enhance the efficiency of data fusion.

The second main research topic this thesis focuses on is search result diversification. Search result diversification is widely being studied as a way of tackling query ambiguity. Instead of trying to identify the "correct" interpretation behind a query, the idea is to make the search results diversified so that users with different backgrounds will find at least one of these results to be relevant to their information need. This thesis examines the hypothesis that data fusion can improve performance in terms of diversity metrics by promoting aspects that are found in disparate ranked lists, to the top of the fused list. We propose a new data fusion method, called diversified data fusion for search result diversification. Based on *latent Dirichlet allocation* (LDA), this method operates on documents in the result lists to be fused, whether the result lists have been diversified or not. It infers latent topics, their probabilities of being relevant and a multinomial distribution of topics over the documents being fused. In addition, in both search result diversification and personalized web search, an issued query is often viewed as an incomplete expression of a user's underlying need. Although different, search result diversification and personalization are not incompatible and do not have mutually exclusive goals. We study the problem of personalized diversification of search results via supervised learning, with the goal of enhancing both diversification and personalization performance. The ideas are two-fold. Search results generated by diversification techniques should be more diverse when a user's preferences are unrelated to the query. Likewise, personalization can improve the effectiveness of aspect weighting in diversification, by favoring query

interpretations which are predicted to be more related to each specific user.

The results in this thesis show how both our proposed data fusion methods and the proposed search result diversification methods improve retrieval performance and how data fusion and diversification relate to each other. The insights from the work in this thesis may be used to improve retrieval performance for a range of tasks in information retrieval.

Samenvatting

Data fusion en diversificatie van zoekresultaten zijn twee essentiële onderzoeksgebieden in information retrieval. Het eerste onderwerp van dit proefschrift is data fusion. Data fusion methoden combineren zoekresultaten van verschillende zoekalgoritmes om zo een beter geordende lijst van resultaten te verkrijgen. Bovengenoemde zoekalgoritmes kunnen heel verschillend zijn. Ze kunnen bijvoorbeeld verschillen in de manier waarop ze zoekvragen of documenten representeren. Veel state-of-the-art data fusion methodes zijn gebaseerd op de aanname dat alleen documenten die hoog gerangschikt worden door veel zoekalgoritmes waarschijnlijk relevant zijn. Als een document dus maar voorkomt in de resultaten van één algoritme en bovendien met een lage score, dan zal het in de uiteindelijke gecombineerde lijst van resultaten ook laag gerangschikt zijn. Wanneer we data fusion bijvoorbeeld toepassen voor het doorzoeken van microblog posts, dan kan deze aanname tot onbevredigende resultaten leiden. Wij ontwikkelen twee nieuwe data fusion algoritmes die nieuwe informatie gebruiken ten opzichte van traditionele methoden. De eerste van deze twee methodes legt de nadruk op het gebruik van temporele informatie, d.w.z. het tijdstip van publicatie van microblog posts. In eerder onderzoek wordt vaak impliciet of expliciet aangenomen dat de score van een document dat niet wordt gevonden door een zoekalgoritme op nul moet worden bepaald, een aanname die wij ter discussie stellen. In ons tweede data fusion algoritme gebruiken we latente variabelen om de scores van dergelijke documenten te schatten. Daarbij laten we ons leiden door de intuïtie dat documenten die lijken op documenten die wèl voorkomen in een lijst ook ongeveer dezelfde score zou moeten worden toegekend. Tot nu toe heeft men bij data fusion vooral gewerkt met de scores of rangorde van documenten in de lijsten van de verschillende zoekalgoritmes, zonder een diepgaande analyse van de mate waarin documenten op elkaar lijken. In deze dissertatie introduceren we een derde data fusion methode gebaseerd op variëteiten. De methode construeert variëteiten voor documenten gebaseerd op de mate waarin deze documenten op elkaar lijken. Documenten met lage scores die in dezelfde variëteit liggen als documenten met hoge scores worden opgewaardeerd. Om de efficiëntie van data fusion te verhogen wordt hierbij gebruik gemaakt van top-k documenten als referentiepunten.

Het tweede hoofdonderwerp van dit proefschrift is diversificatie van zoekresultaten. Hier wordt veel onderzoek naar gedaan, met name als een manier om om te gaan met de ambiguïteit van zoekvragen. In plaats van te proberen zoekvragen "correct" te interpreteren wordt er diversiteit aangebracht in de zoekresultaten, zodat mensen met een verschillende achtergrond en bedoeling tenminste één van de resultaten relevant zullen vinden. In dit proefschrift onderzoeken we de hypothese dat data fusion kan helpen om de kwaliteit van zoekresultaten te verbeteren in termen van diversificatie-metrieken. We doen dit door aspecten van de zoekvragen die in zoekresultaten van verschillende algoritmes voorkomen naar voren te brengen in de gecombineerde resultaten. Dit leidt tot een nieuwe data fusion methode voor diversificatie van zoekresultaten. Deze methode, gebaseerd op *Latent Dirichlet Allocation* (LDA), maakt gebruik van de zoekresultaten van alle algoritmes die gecombineerd moeten worden, of deze nu zelf al diversificatie toepassen of niet. Er worden latente onderwerpen gevonden in de te combineren documenten, dan wordt geschat hoe relevant deze onderwerpen zijn voor de zoekvraag. De documenten worden gerepresenteerd door een multinomiale distributie van deze onder-

werpen. In onderzoek naar zowel diversificatie als personalisering van zoekresultaten wordt een zoekvraag vaak gezien als een onvolledige uitdrukking van een onderliggende informatiebehoefte. Hoewel ze verschillen, zijn diversificatie en personalisering niet onverenigbaar. De doelstellingen van beide sluiten elkaar niet uit. In dit proefschrift onderzoeken we het probleem van gepersonaliseerde diversificatie van zoekresultaten. We verbeteren zowel de diversificatie als de personalisatie met een methode gebaseerd op gesuperviseerd leren. De ideeën zijn tweeledig. Zoekresultaten zouden meer diverse moeten zijn als de voorkeuren van de gebruiker ongerelateerd zijn aan de zoekvraag. Omgekeerd kan personalisatie de kwaliteit van de resultaten verbeteren door interpretaties van de query die stroken met de voorkeuren van de gebruiker sterker te benadrukken. De resultaten in dit proefschrift laten zien hoe onze data fusion en diversificatie methoden de kwaliteit van de rangschikking van zoekresultaten verbetert. De resultaten geven ook inzicht in de verbanden tussen data fusion en diversificatie.