2013

Mining and Understanding Online Conversational media

Liangjie Hong
Lehigh University

Mining and Understanding Online Conversational Media

by

Liangjie Hong

A Dissertation
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Computer Science

Lehigh University
May 2013

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Liangjie Hong
Mining and Understanding Online Conversational Media

Date

Professor Brian D. Davison, Dissertation Director, Chair
(Must Sign with Blue Ink)

Accepted Date

Committee Members

Professor Jeff D. Heflin

Professor Roger N. Nagel

Professor Soutir Bandyopadhyay

Acknowledgements

This dissertation is impossible without the help from my advisor Prof. Brian D. Davison. I wish to thank him to provide me with the freedom through my Ph.D. to explore various research problems while giving guidance and stimulating feedback. Prof. Davison taught me a lot of things: how to see the big picture, show to write and prepare submissions, how to present my papers and how to develop a taste for selecting the right problems to work on. Throughout my time at the WUME lab in Lehigh University, I was really impressed with how dedicated Prof. Davison was to the research and to his students, not even mentioning working with us very early in the morning for paper submissions. Prof. Davison is not only an advisor for research, but also a mentor for many aspects in my life, especially for career planning. He also sets a very high bar for his students, both in terms of the quality of work we produce and the potential skills we develop, which I personally benefit greatly from.

Part of this dissertation was done when I was an intern in Yahoo! Labs and LinkedIn. I was very lucky to have an amazing team of colleagues and mentors in both places: Amr Ahmed, Kostas Tsioutsiouliklis, Siva Gurumurthy, Alexander J. Smola, Byron Dom, Marco Pennacchiotti, Ron Bekkerman and Joseph Adler. I deeply thank Kostas for having me onboard in Yahoo! Labs twice as an intern, which was very rare a case for the organization as far as I know, even though for both times we produce high-quality papers. Byron was really insightful about problems and he was always ready to write programs to validate our ideas. It was a lot of fun to worked with him on white boards to derive equations. Amr is definitely a great mentor. He is always encouraging even when we missed the first deadline of our paper submission (due to my fault). One striking thing I know from Amr is that he was sometimes discussing problems with me in Starbucks (Once I remembered was a Friday night). His dedication to research has set a very high example for me. I learn a lot of Bayesian inference from Amr. For Alex, he is a resource of research

ideas. Alex provides a good role model for researchers. One extraordinary example I have is that the introduction part of our WWW 2012 paper was almost re-written by him during his time in an airport in Germany. Researchers should be 24/7, in Alex's sense. Siva is a great collaborator and I really enjoy the afternoon walk we took in Yahoo!'s campus to discuss problems and implementation details. He is practical yet innovative. During my time in LinkedIn, Ron gave me a lot of freedom to explore things. Ron is fun and energetic. He provided tremendous support for our project and devoted a lot of time to work with me on our paper.

My friends in Lehigh University deserve a special thank. I would like to thank Xiaoguang Qi and Lan Nie. As senior members in the lab, they provided endless help for my life and research projects. I also greatly value the collaboration with WUME lab members: Dawei Yin, Ovidiu Dan, Zaihan Yang and Jian Wang, as well as thoughtful discussions with Na Dai. I will remember a lot of nights we spent in WUME lab office in Packard lab with Dawei and Zaihan. Ovi and I went to India for WWW 2011 and we won the best poster paper together, which was a wonderful experience in my life. I also would like to thank my roommate Qian Wu, who is also a Ph.D. student in Lehigh for three years of sharing an apartment with me. It was a lot of fun.

Finally, I am deeply indebted to my dear mother and father for their love and strong support during my graduate study. They have both been with me through the whole process with its ups and downs. I would like to thank my dear wife Liuqing for her strong support through all these years to my study and career. Words can hardly express how fortunate I am for having her in my life. We married in the final year of my Ph.D. study, which strengthened an eight-year long journey of love from the year of 2003. This thesis is dedicated to them.

Most of the work presented in this dissertation was supported in part by grants from the National Science Foundation under awards IIS-0545875 and IIS-0545875, as well as

an equipment grant from Sun Microsystems. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Dedication

To my beloved wife, Liuqing Zheng
To my parents, Xin Zeng & Xiangming Hong

Contents

List of Tables

List of Figures

Abstract

The social network-enhanced Web has become increasingly important. With a wide spectrum of social services such as blogs, wikis, online forums, social network services and community question answering portals, individuals can produce, consume and share information through rich user interactions. These interactions include conversations, annotations and resource sharing, enabling faster and wider dissemination and development of information at a large scale. In addition, the recent popularized micro-blogging services such as Twitter and Tumblr have revolutionized the Web to a more synchronized world, opening opportunities for users around the world with various cultural backgrounds to generate and propagate information in "real-time". Conversations as a scientific field have been studied for decades. Traditional research related to conversations has been considered by a variety of disciplines including linguistics, sociology, anthropology, psychology, communication studies and translation studies, each of which is subject to its own assumptions, dimensions of analysis, and methodologies. One major characteristic of traditional research on conversations is that most previous classic studies were based on surveys, field research, small scale datasets and sometimes depend on a detailed inspection of tape recordings or transcriptions made from such recordings. While some theories and methodologies developed in these areas became the foundations for modern analysis of conversations (e.g., the ones based on computational linguistics and information retrieval), most of them cannot be directly applied to online settings due to their qualitative nature and also due to some of their case-by-case style of studies that cannot be scaled to the amount of data online. In addition, since such research was conducted prior to the time of popularity of the Internet, the conclusions and results obtained through these methods are also needed to be re-verified in the new era as well.

Although a large amount of research has been made in mining and understanding online conversational media, some practical problems remain unanswered. First of all, when

facing a large amount of socially generated content, users simply cannot consume it in an effective and efficient way, leading to the problem of information overload. On the other hand, it is difficult for a user to obtain information distributed outside of their social circle, even though it might match their interests, leading to the problem of information shortage. Users may spend a significant amount of time to filter and search relevant information in such platforms. In general, the problem can be considered as information filtering in online conversational media. One of the central challenges to information filtering is to track users' interests. The assumption is that if we can understand them perfectly, most relevant and fresh information can be selected from the ocean of items and presented to users. The key ingredient of tracking users' interests for online conversational media is to understand the content generated by users, usually modeled as topical distributions, as well as rich interaction data.

In this dissertation, we will discuss both information filtering and topic/interest tracking as they are two important problems in online conversational media, in a principled way. On one hand, we will demonstrate how we develop new approaches to achieve the state-of-the-art performance in each direction. On the other hand, we will also discuss the relationships between these two directions and show how they can indeed link with each other. We link two directions of mining and understanding online conversational media as a dual relationship of data analysis in online conversational media and demonstrate that how they benefit from the development of each other. This dissertation can be used as a guideline for readers who are interested in data analysis in social media in general.

Chapter 1

Overview

In this chapter, we briefly review the history of online conversational media and how it dramatically contrasts with traditional Web . Then, we outline two fundamental research challenges of mining and understanding online conversational media: 1) information filtering and 2) modeling users' interests/topics, revealing that they are two sides of the same coin and have significant interplay with each other. More specifically, in this chapter, we argue that the core of information filtering is to understand what users' interests are and what they are talking about in online conversational media as user generated content is a key component in such media sources. In the latter part of the chapter, we outline a series of steps, which consists of the main part of this dissertation, to achieve better performance by incorporating interests/topics tracking components into the filtering or recommendation system. In addition, as topic tracking is so important, we will introduce its unique challenges in online conversational media and how we can cope with them by discussing the contributions of this dissertation on this direction.

1.1 Online Conversational Media

The social network-enhanced Web has become increasingly important. With a wide spectrum of social services such as blogs, wikis, online forums, social network services and community question answering portals, individuals can produce, consume and share information through rich user interactions. These interactions include conversations, annotations and resource sharing, enabling faster and wider dissemination and development of information at a large scale. In addition, the recent popularized micro-blogging services such as Twitter and Tumblr have revolutionized the Web to a more synchronized world, opening opportunities for users around the world with various cultural backgrounds to generate and propagate information in "real-time".

The traditional Web is dominated by static pages, created by various organizations and users, serving the role of publishing information. User interactions rarely take place in such environments as the technologies available at that time did not support rich user interactions well. For example, the pure HTML standard at that time did not allow users to easily post new content to a static page. Also, the presentation of web pages cannot be easily modified on the fly. In addition, a service relying on users' feedback is usually created by heavy-weighted Common Gateway Interfaces (CGIs), which is much more difficult to be written and deployed, compared to today's web frameworks. All these technological issues hindered the Web 1.0 to be able to serve as a platform for users' communications. Some may argue that users' interactions do exist in the early Web era. For instance, mailing lists and newsgroups are such examples where users exchange ideas and discuss issues through emails and other types of clients, rather than Web browsers. Thus, compared to Web 2.0 paradigms, the dynamic and scale of these interactions is constrained.

Differing from the traditional Web, today's socially-enabled Web not only embraces

newly developed technologies to allow better user interactions on the Web but also opens new opportunities for mobile devices to access user generated data, including text, images, audio and video. One interesting type of data is conversations between users. Here, conversations are defined informally as information exchange between users in an informal way. Indeed, on a socially-enabled Web, users may have a variety of channels to initialize conversations, with different purposes in mind. For instance, question answering portals have attracted users to interact on specific problems and open issues while forums are generally places for more free discussions. Also, Facebook might be for talks between friends and acquaintances but LinkedIn is for professional opinions. These online conversations, usually accompanied by rich metadata, play a major role in today's dynamic Web ecosystems. Online conversations, though taking place in different platforms and services, share a number of unique characteristics. First of all, most of them are *informal* interactions between users, indicating that grammar and syntax of them might have errors. Second, although multimedia data is prevalent on the Web nowadays, a overwhelming amount of conversational data is still textual, providing a significant opportunity for deeper understanding. Third, many conversations are shared and propagated through users' connections in online social networks (e.g., Facebook, Twitter, Weibo, etc.), making a much wider impact than other formats. Finally conversations are used to shape and characterize users' interests, serving as "footprints" for users, which allows online media companies and advertising agencies to target their audience in a precise manner.

The changes and transformations of the Web, distinguished by online conversations with social media, have been bringing tremendous opportunities as well as challenges for multiple research communities. On one hand, researchers are facing an amount of user interaction data that has never been explored before, which reveals many details of such interactions that cannot be captured through traditional studies such as surveys and field studies. On the other hand, practical and theoretical challenges remain for

better understanding and predicting users' behaviors and relations in online conversational media. In particular, in order to perform reasoning in a large scale and handle inherent uncertainty, computational methods and probabilistic modeling become central tools to tackle such predictive problems.

Conversations as a scientific field have been studied for decades. Traditional research related to conversations has been considered by a variety of disciplines including linguistics, sociology, anthropology, psychology, communication studies and translation studies, each of which is subject to its own assumptions, dimensions of analysis, and methodologies. For example, linguistics mainly focuses on understanding the discourse structures and patterns of conversations at various levels such as syntax, lexicon, style, rhetoric, meanings, speech acts and other aspects of interaction (e.g., [75, 32]). While these structures cannot be easily ignored to understand conversations, they might be at a too detailed level. In addition, as discussed before, online conversations are usually not well-formed and so linguistic analysis may face difficulties to uncover these structures. While in the sociology context, researchers try to identify the organization of conversations (e.g., "turn-taking" [173]) and their impact on human status and the corresponding interactions [14, 102]. For instance, some studies reveal how social relations, identity, knowledge and power are reflected through conversations and how conversations are performed in informal settings and institutional environments (e.g., classroom, court, conference, etc.). However, one major characteristic of traditional research on conversations is that most previous classic studies were based on surveys, field research, small scale datasets and sometimes depend on a detailed inspection of tape recordings or transcriptions made from such recordings. While some theories and methodologies developed in these areas became the foundations for modern analysis of conversations (e.g., the ones based on computational linguistics and information retrieval), most of them cannot be directly applied to online settings due to their qualitative nature and also due to some of their case-by-case style of studies

that cannot be scaled to the amount of data online. In addition, since such research was conducted prior to the time of popularity of the Internet, the conclusions and results obtained through these methods are also needed to be re-verified in the new era as well. For instance, in 1969, Jeffrey Travers and Stanley Milgram [189] conducted the famous "small world experiment" by sending mail among real persons to see whether small networks of people exist in the society. With the experimental results in their hand, the authors conjectured that a short distance between any pair of persons may exist, without any large scale validations. Today, this kind of conjectures can be easily verified on massive online conversational media such as Facebook and Twitter. For instance, in a recent study, Backstrom et al. [15] observed that more than 721 million users on Facebook share a 4.74 degree of distance on average between users, which is even smaller than what researchers might have expected. This is a representative example of how current research on similar topics differs from traditional research not only on the conclusions but also on the scale and the methods. In general, existing research on conversations needs to be adapted to a larger setting, not focusing on a small number or a group of people but on the tremendous number of online users. New approaches, which are equipped by up-to-date high performance computing architectures such as Google's MapReduce [58], to analyze the vast volume of conversational data generated in the Internet era are deeply desired.

Apart from classical work on conversations, current research in social computing, information retrieval, web mining, human computer interaction and computational linguistics, is starting to address many similar problems related to conversational environments, such as how to find useful information, how to understand user interactions and how to facilitate knowledge discovery and creation. For instance, machine learning methods have been developed to analyze email to understand speech acts and activities [52, 36, 37, 124]. By analyzing thousands of email messages, these methods discover striking patterns in complex business environments like Enron, moving beyond the traditional research which is

based on a small number of conversations between a handful of individuals. Similar work has also been conducted on newsgroups to better classify messages and reconstruct conversations and user interactions [203, 202]. For more social platforms, methods are proposed to better model conversation–like in blogs [68, 13] and discussion boards or online forums [133, 176, 69]. In addition, researchers have studied methods [212, 4] to better identify high quality content in community–based question answering portals where users communicate with each other while the solutions and comments are generated in a conversational fashion. Furthermore, a variety of studies have been conducted to group interleaving text streams into conversations [200, 70] and understand the topics within them [66, 44]. Also, a lot of recent attention has been paid to the phenomenon of microblogging services and trying to understand how events and messages are spread in such a highly interactive user network [46, 59, 177].

1.2 Research Challenges

Information filtering and topic/interest tracking are not two separate domains for online conversational media but they are indeed two sides of the same coin. Information filtering can be treated as an important application of topic/interest tracking while the latter is the central component of the former. We will show later in this dissertation that these two tasks can benefit from each other and can be modeled simultaneously.

1.2.1 Information Filtering

Although a large amount of research has been made in mining and understanding online conversational media, some practical problems remain unanswered. For instance, many online conversational services such as, Twitter, Facebook, LinkedIn and Yahoo! Answers, serve as platforms for users to obtain fresh and relevant information. Some may require

users to actively search and browse among the repository of items according to their information needs while others may allow users to subscribe to feeds from their peers to obtain fresh updates. Nevertheless, the sheer amount of content generated by users is causing two issues that prevent it from being sufficiently relevant to users; deteriorating user experience and engagement. First of all, when facing a large amount of socially generated content, users simply cannot consume it in an effective and efficient way, leading to the problem of *information overload*. On the other hand, if users subscribe to their social connections, information for a user is usually limited in scope to the their connections. Thus, it is difficult for a user to obtain information distributed outside of their social circle, even though it might match their interests, leading to the problem of *information shortage*. Users may spend a significant amount of time to filter and search relevant information in such platforms. From the perspective of service providers, it is also very important to understand how users interact with the systems through a variety of actions such as re-posting, replying, commenting and clicks. In general, the problem can be considered as **information filtering** in online conversational media. This problem has some significant challenges. First, although the number of items generated by users in services is huge, a particular user will interact with few of them, making the interaction data sparse. Second, new users and new content items flow into the system continuously. Thus, the "cold start" problem tends to be severe in these social platforms, compared to traditional information systems. In addition, a tremendous amount of content is rich yet noisy. Simple information retrieval or topical modeling techniques may not be sufficient to capture users' interests. To address both the problemof information overload and information shortage, social media monitoring systems are built to filter and recommend content items to users based on numerous signals. This area has recently attracted close attention of academic and industrial research communities.

The task of information filtering can be approached from various perspectives. From

the Information Retrieval (IR) perspective, constructing personalized information results can be cast into the classic ranking problem: the task is to rank items by descending order of user interest. It may be true that some existing IR techniques could be potentially applied to social information filtering. However, user's interests in online conversational media are not represented in terms of a search query. Instead, queries are implicit and have to be inferred. The absence of a search query distinguishes the information filtering problem under social context from many classic IR tasks. In addition, social information needs are more diversified compared to traditional IR scenarios. Although traditional IR tools do not appear to be directly applicable to ranking in online conversational media, some of recently developed *learning to rank* approaches are very appealing to be used in the new setting. From the perspective of Recommender Systems (RecSys), building relevant list of information items can be viewed as recommending relevant items to users. Thus, many collaborative filtering techniques are applicable to the task of social stream ranking. However, as discussed before, online conversational media services are much more dynamic than traditional information systems: many new items can be pushed into the system every second. Therefore, the cold start problem becomes even more severe in such platforms. The traditional collaborative filtering paradigm needs to be adjusted in the new environment.

1.2.2 User Interest Profiling

One of the central challenges to information filtering is to track users' interests. The assumption is that if we can understand them perfectly, most relevant and fresh information can be selected from the ocean of items and presented to users. The key ingredient of tracking users' interests for online conversational media is to understand the content generated by users, usually modeled as topical distributions, as well as rich interaction data.

Topic modeling is a fundamental problem in text mining and many techniques have been developed for such purposes. Topic models have been applied to numerous text corpora to find latent topics to help people visualize and understand the themes of the corpora, with little or no supervision information. However, as discussed before, online conversational media is full of dynamic and noisy data, presenting a number of significant challenges for interest tracking. For instance, traditional topic modeling usually targets a static text corpus where the size of vocabulary is limited and the content of documents in the corpus is usually well-written. On the contrary, topics in online conversational media emerge and vanish over time. These dynamics cannot be easily captured by standard tools. Also, in online conversational media, we not only wish to discover topics from a single data corpus but also compare and track topics from multiple data services (e.g., Twitter, news sources, forums). Thus, it might be desirable for topic modeling techniques to be extended onto multiple data sources and model them simultaneously. In addition, another drawback of these existing models is that most of them are general purpose with which no real tasks are explicitly associated. Therefore, it might be difficult to employ these models in real-world applications, such as for the problems of tracking trends and predicting popularity of keywords. As a result of the lack of a particular task, there is also no consensus on how these models should be evaluated and compared. Although perplexity is usually used for evaluating predictive power of models, it is not designed for any real-world tasks and might not reflect their performance anyway.

Last but not least is the prevalence of meta data associated with the generated content. Some of the meta data provides indispensable information on how topics should be formed and analyzed. For instance, meta data like geographical locations raises new research questions like 1) How is information created and shared in different geographic locations? What is the inherent geographic variability of content? 2) What are the spatial and linguistic characteristics of people? How does this vary across regions? 3) What is a good

model for human mobility? Can we discover patterns in users' usage of micro-blogging services? All of these questions cannot be easily answered without tailoring existing models to be aware of the corresponding meta data. Because of these drawbacks of existing topic modeling techniques, when applying to conversational media, the research community is calling for new approaches and dedicated extensions.

1.3 Contributions

In this dissertation, we will discuss both information filtering and topic/interest tracking as they are two important problems in online conversational media, in a principled way. On one hand, we will demonstrate how we develop new approaches to achieve the state-of-the-art performance in each direction. On the other hand, we will also discuss the relationships between these two directions and show how they can indeed link with each other. For information filtering, we will show that:

- In community–based question answering portals, we explore the problem of filtering question answering content from discussion boards and divide it into two subtasks: identifying question–related first posts and finding potential answers in subsequent responses within the corresponding threads. We address both subtasks as classification problems and choose several content–based and non-content based features and carefully compare them individually and also in combinations. We do not use any service or dataset–specific heuristics or features (like the rank of users) in our classification model; therefore our approach should be usable in any discussion board. We compare our approach with previous methods and show significant improvements in experimental results.

- For the problem of filtering messages in micro-blogging services, we first tackle a broader version of the problem, which is to identify popular messages. We treat

it as a classification task. We train classifiers with positive and negative examples of messages which will be retweeted in the future which are shared in the future. To build such classifiers, we investigate a wide spectrum of features to determine which ones can be successfully used as predictors of popularity, including the content and topical information of messages, graph structural properties of users, temporal dynamics of popular messages and meta-information of users and messages as well. Our experiments are conducted on two massive real-world datasets and the results suggest that we can successfully predict whether a message will be popular or not and its volume with good predictive performance. The work tries to answer the following questions: 1) What features are useful for predicting popular messages? 2) Are the features for low volume popular messages the same as the ones with high volume? 3) Are the popular messages predicted from our method "legitimate messages", or just spam?

- To filter information for each user, we study a surrogate problem of predicting whether a user will take actions towards a message in micro-blogging environment. Our method can be easily extended to model multiple types of users' decisions as well. We use a state-of-the-art recommendation model, Factorization Machines `FM` [165], to model user decisions and user-generated content simultaneously. In particular, we propose Co-Factorization Machines (`CoFM`), which deal with two (multiple) aspects of the dataset where each aspect is a separate `FM`. This type of model can easily predict user decisions while modeling user interests through content at the same time. With this tool, we apply Factorization Machines to text data with constraints. Thus, the resulting method can mimic state-of-the-art topic models and yet benefit from the efficiency of a simpler form of modeling. For user decision modeling, we compare a number of ranking-based loss functions and introduce the newly proposed WARP loss [190] into the context of information filtering and recommendation. We

apply our proposed methods to the problem of modeling personal decision making in Twitter and explore a wide range of features, revealing which types of features contribute to the predictive modeling and how content information can help with the prediction.

For tracking users' interests and topics, we address many problems mentioned above and extend existing techniques to achieve state-of-the-art performance in a number of tasks in online conversational media. In particular, we have developed the following models:

- In order to track topic trends, rather than building a general-purpose model, we propose a new type of topic model incorporating the volume of terms into the temporal dynamics of topics and directly optimize for the task. Unlike existing models in which trends are either latent variables or not considered at all and thus are difficult to apply in practice, we combine state-space models with term volumes in a supervised learning fashion which enables us to effectively predict volumes in the future, even without new documents. In addition, it is straightforward to obtain the volumes of latent topics as a by-product of our model, demonstrating the superiority of utilizing temporal topic models over traditional time-series tools (e.g., autoregressive models) to tackle this kind of problem. The proposed model can be further extended with arbitrary word-level features which are evolving over time. We present the results of applying the model to two datasets with long time periods and show its effectiveness over non-trivial baselines.

- For modeling multiple social data sources simultaneously, we extend topic models by allowing each text stream to have both local and shared topics. Also, we associate each topic with a time-dependent function that characterizes its popularity over time. By combining the two models, we effectively model temporal dynamics of multiple correlated text streams in a unified framework. The new model can easily discover

common and uncommon topics from multiple text collections with their temporal dynamics. The proposed method is a simple and potentially scalable algorithm for mining temporal topics. We mined interesting results from Yahoo! News and Twitter obtained by applying our model.

- For geographical topic modeling, we propose a model that is both flexible enough to embed all reasonable components of content and geographical locations, as well as user preference modeling. Moreover, it scales to real-world datasets to handle millions of documents and users. We address the problem of modeling geographical topical patterns on Twitter by introducing a novel sparse generative model. It utilizes both statistical topic models and sparse coding techniques to provide a principled method for uncovering different language patterns and common interests shared across the world. Our approach is vital for applications such as user profiling, content recommendation and topic tracking and the method can be easily extended in a number of ways. We show that interesting topics can be identified by the model and we demonstrate its effectiveness on the task of predicting locations of new messages and outperforms non-trivial baselines.

In later chapters, we link two directions of mining and understanding online conversational media as a dual relationship of data analysis in online conversational media and demonstrate that how they benefit from the development of each other. This dissertation can be used as a guideline for readers who are interested in data analysis in social media in general.

1.4 Organization

In Chapter 2, we discuss the basic knowledge of both information filtering and topic tracking. In particular, we put all the models and methods discussed in this dissertation

under the umbrella of latent space/factor models. We will discuss how they can be viewed as the same type of model [18, 181] with different objective functions and different choices of underlying probabilistic distributions. From Chapter 3 to Chapter 5, we discuss several information filtering systems are built for online conversational media and how they evolve over time. By demonstrating the details of these methods, we show that our proposed methods can significantly outperform state-of-the-art algorithms .

- In Chapter 3, we overview of how simple information filtering systems can be built by investigating filtering possible answers to questions in community-based question anwering portals. While we are tackling some important problems in online conversational media, no users' interests/topics are modeled and additionally, no latent structure or insights are discussed through the predictive models we built for the task. Certainly, we demonstrate some limitations and shortcomings of normal machine learning approaches in online conversational media. The material in this chapter was published in SIGIR 2009 and SIN 2009 [93, 97].

- In Chapter 4, we describe a framework to study how we can build an information filtering system regardless of providing personal recommendations. In particular, we predict the popularity of Twitter messages and study the effectiveness of a wide range of features. The material in this chapter was published in WWW 2011 [92].

- In Chapter 6, we take one step further to study how personal information filtering system can be built by investigating how social update streams in LinkedIn. The material in this chapter was published in SIGIR 2012 [91].

- In Chapter 5, finally, we describe a system can both consider information filtering and topic tracking simultaneously by proposing a Co-Factorization Machine. We also discuss how different ranking objective functions behave in a large real-world data. The material in this chapter was published in WSDM 2013 [96].

- In Chapter 7, we focus on the problem of whether topical information is really helpful in online conversational media and if so, how can we utilize it. More specifically, we demonstrate how topic models can be trained under the context of online conversational media. This is a bridge between the first part and the second part of the dissertation to give a path of how topic models can be utilized in general. The material in this chapter was published in SOMA 2010 [94].

From Chapter 8 to Chapter 10, we address the second direction of the dissertation, which is to uncover hidden patterns from online conversational media.

- In Chapter 8, we address two essential problems of modeling topics in online conversational media, which is to tackle multiple time-varying data sources. The material in this chapter was published in KDD 2011 [95].

- In Chapter 9, we explore the idea of directly modeling terms' volume in the context of topic modeling. The approach introduced can be utilized in tracking trending topics more precisely than previous work. The material in this chapter was published in KDD 2011 [98].

- In Chapter 10, we address the problem of modeling geographical language variations in online conversational media. The material in this chapter was published in WWW 2012 [90].

In Chapter 11, we will conclude the dissertation and discuss future directions.

Chapter 2

Foundations and Preliminaries

In this chapter, we discuss certain aspects of the foundations of statistical machine learning. The goal is to provide background knowledge to later chapters. More specifically, we focus on the theories and techniques that influence the models and methods introduced in this dissertation.

We start the discussion on the definition of exchangeability and its implications in section 2.1, especially how it provides the arguments for linear models and latent factor models. In section 2.2, we discuss several linear models used in this dissertation and their unified form. In section 2.3, we discuss how all matrix-factorization-based models can be treated under a same framework and how it can be extended to multiple entities, leading to a number of similar proposed frameworks, such as co-factorization methods.

In this dissertation, matrices are denoted by capital **bold** letters, $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$. Elements, rows and columns of a matrix are denoted $\mathbf{X}_{i,j}, \mathbf{X}_{i\cdot}, \mathbf{X}_{\cdot j}$. Vectors are denoted by lower case **bold** letters, and are assumed to be **column** vectors. Conventionally, we use $\mathbf{X} \in \mathbb{R}^{M \times N}$ to denote a design matrix or a feature matrix where each row $\mathbf{X}_{i\cdot} \in \mathbb{R}^N$ is a row-vector, representing a data instance. Each element in $\mathbf{X}_{i\cdot}$ is a feature, characterizing the data instance in a certain way. We use $\mathbf{y}$ to represent the response vector where $\mathbf{y}_i$ could be a

binary label or a real-valued response. If we have two matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{A} \circ \mathbf{B}$ denotes the matrix inner product and $\mathbf{A} \odot \mathbf{B}$ denotes the element-wise (Hadamard) product.

2.1 Preliminaries

In this sub-section, we provide a brief introduction on several fundamental definitions and theorems on statistical modeling, providing justification to the linear models and latent variable models used in this dissertation. We start our discussion with the exchangeability definition on a sequence of random variables, namely an array of random variables in sub-section 2.1.1. Then, we extend the discussion to two dimensional scenarios, matrices in sub-section 2.1.2. In sub-section 2.1.3, we review the basics about exponential families and Bregman divergences, which are used to formalize the unified view of matrix factorization and collective matrix factorization.

2.1.1 1D Exchangeability & De Finetti's Theorem

The following definition and the theorem is introduced in [9] while here, we follow the presentation of [85].

Definition 1. *(Infinite Exchangeability). We say that $(x_i, x_2, \cdots, x_n)$ is an infinitely exchangeable sequence of random variables if, for any n, the joint probability $P(x_1, x_2, \cdots, x_n)$ is invariant to permutation of the indices. That is, for any permutation π,*

$$P(x_1, x_2, \cdots, x_n) = P(x_{\pi_1}, x_{\pi_2}, \cdots, x_{\pi_n})$$

A key assumption of many statistical analyses is that the random variables being studied are independent and identically distributed (iid). Note that iid random variables are always infinitely exchangeable. However, infinite exchangeability is a much broader

concept than being iid; an infinitely exchangeable sequence is not necessarily iid. Now, we can review de Finetti's theorem below:

Theorem 1. *(de Finetti's Theorem). A sequence of random variables* $(x_1, x_2, \cdots, x_n)$ *is infinitely exchangeable* ***if and only if****, for all* n*,*

$$P(x_1, x_2, \cdots, x_n) = \int \prod_{i=1}^{n} P(x_i \,|\, \theta) P(\theta) d\theta$$

for some measure P *on* θ*.*

Theorem 1 implies many important results in probabilistic modeling. For instance, it provides justification for using parameters to characterize data, which is indeed what both linear model and latent variable models are trying to do.

2.1.2 2D Exchangeability & Aldous' Theorem

Going beyond the exchangeability definition on one dimensional arrays, we can extend it to two dimensional matrices. The following definition and theorem is introduced by Aldous [10] but we follow a simpler discussion from [85].

Definition 2. *(2D Array/Matrix Exchangeability). We say that matrix* $\mathbf{X}$ *is an row-column exchangeable matrix if, for any* i *and* j*, the joint probability* $\prod_{i,j} P(\mathbf{X}_{i,j})$ *is invariant to permutation of the indices. That is, for any permutation* π*,*

$$P(\mathbf{X}_{i,j}) = P(\mathbf{X}_{\pi(i),\pi(j)})$$

Exchangeability in this case can be interpreted as saying that the row labels and the column labels carry no information about $\mathbf{X}$. The analogue of Theorem 1 in 2D/Matrix scenarios is the following theorem:

Theorem 2. *(Aldous' Theorem). If* $\mathbf{X}$ *is row-column exchangeable, then there exists a function* g *and independent uniformly distributed random variables* $\mu, \{u_1, \cdots, u_m\}, \{v_1, \cdots, v_m\}$, *and* $\epsilon_{i,j}$ *such that:*

$$P(\mathbf{X}_{i,j}) = g(\mu, u_i, v_j, \epsilon_{i,j})$$

The Theorem 2 says that any statistical model of a row-column exchangeable matrix can be parametrized by a global effect μ, a row effect u_i, a column effect v_j, and a dyadic effect $\epsilon_{i,j}$ (an interaction term between i and j). Moreover, the dyadic effects are independent of one other. It should be noted that Theorem 2 does not imply any particular form of g.

2.1.3 Exponential Families and Bregman Divergence

Consider the exponential family with natural parameter $\boldsymbol{\theta} \in \mathbb{R}^d$; then the exponential family probability density function can be written as:

$$P(\mathbf{x} \,|\, \boldsymbol{\theta}) = \exp\Big(< \mathbf{x}, \boldsymbol{\theta} > + \log P_0(\mathbf{x}) - F(\boldsymbol{\theta}) \Big) \tag{2.1}$$

where $P_0(\mathbf{x})$ is a base measure, independent of the parameters and $F(\boldsymbol{\theta}) = \log \int P_0(\mathbf{x}) \exp(< \mathbf{x}, \boldsymbol{\theta} >) \, d\mathbf{x}$ is the log-partition function, ensuring the distribution is normalized. In order to better link to Bregman divergence, we use the following variant as the definition of the exponential family:

Definition 3. *A parametric family of distribution* $\psi_F = \{P_F(\mathbf{x} \,|\, \boldsymbol{\theta}) : \boldsymbol{\theta}\}$ *is a regular exponential family if each density* P_F *can be expressed as the following canonical form:*

$$\log P_F(\mathbf{x} \,|\, \boldsymbol{\theta}) = < \mathbf{x}, \boldsymbol{\theta} > + \log P_0(\mathbf{x}) - F(\boldsymbol{\theta})$$

Note that a distribution in ψ_F is uniquely identified by its natural parameters.

Next, we introduce Bregman divergence and its relationship to exponential families.

Definition 4. *(Generalized Weighted Bregman Divergence) For a closed, proper, convex function $f : \mathbb{R} \to \mathbb{R}$ and constant weight matrix $\mathbf{W} \in \mathbb{R}_+^{M \times N}$, the generalized weighted Bregman divergence is:*

$$\mathbb{D}_F\Big(\mathbf{\Theta} \,||\, \mathbf{X}, \mathbf{W}\Big) = \sum_{i,j} \mathbf{W}_{i,j}\Big(F(\mathbf{\Theta}_{i,j}) + F^*(\mathbf{X}_{i,j}) - \mathbf{X}_{i,j}\mathbf{\Theta}_{i,j}\Big)$$

where F^* is the convex conjugate defined as $F^*(\mu) = \sup_{\Theta \in dom\, F}\Big[\Theta \circ \mu - F(\Theta)\Big]$.

This definition differs from several traditional ones like [18] as here F is allowed to be non-differentiable. If F is additionally differentiable, $\nabla F = f$ and $\mathbf{W}_{i,j} = 1$, the Definition 4 is equivalent to the standard definition:

$$\begin{aligned}\mathbb{D}_F\Big(\mathbf{\Theta} \,||\, \mathbf{X}, \mathbf{W}\Big) &= \sum_{i,j} F^*(\mathbf{X}_{i,j}) - F^*(f(\mathbf{\Theta}_{i,j})) - \nabla F^*(f(\mathbf{\Theta}_{i,j}))(\mathbf{X}_{i,j} - f(\mathbf{\Theta}_{i,j})) \\ &= \mathbb{D}_{F^*}(\mathbf{X} \,||\, f(\mathbf{\Theta}))\end{aligned}$$

Generalized Bregman divergences are important because they include many common separable divergences, such as squared loss, $F(x) = \frac{1}{2}x^2$, and KL-divergence, $F(x) = x \log x$.

There is a close relationship between Bregman divergences and regular exponential families through:

$$\log P_F(\mathbf{x} \,|\, \boldsymbol{\theta}) = \log P_0(\mathbf{x}) + F^*(\mathbf{x}) - \mathbb{D}_{F^*}(\mathbf{x} \,||\, f(\boldsymbol{\theta})) \tag{2.2}$$

where the $f(\boldsymbol{\theta}) = \nabla F(\boldsymbol{\theta})$. More discussions can be found in [53].

2.2 Linear Models

Linear models are used and extended heavily for a wide range of tasks. Here, we review three fundamental linear models in this section: 1) linear regression, 2) logistic regression and 3) support vector machines. Note that these three models are usually used in different scenarios. For instance, linear regression is mostly used for regression problems while logistic regression is very effective in binary classification problems. On the other hand, support vector machines can be adopted into both regression and classification problems.

Linear Regression: Linear regression starts with an assumption that the estimation of the response $\mathbf{y}_i$ for the data instance can be modeled through a linear function as:

$$\hat{\mathbf{y}}_i = w_0 + \sum_{j=1}^{N} \mathbf{w}_j \mathbf{x}_{i,j} \tag{2.3}$$

where $\mathbf{w} \in \mathbb{R}^N$ is a vector of regression coefficients and w_0 is a bias term. Equation 2.3 can be re-written into:

$$\hat{\mathbf{y}}_i = w_0 + < \mathbf{w}, \mathbf{x}_i > \tag{2.4}$$

where $< . >$ denotes the dot product.

Logistic Regression: Logistic regression models the conditional probability $P(\mathbf{y}_i \,|\, \mathbf{w}, \mathbf{x}_i)$ as:

$$P(\mathbf{y}_i \,|\, \mathbf{w}, \mathbf{x}_i) = \frac{1}{1 + \exp(-\mathbf{y}_i \mathbf{w}^T \mathbf{x}_i)} \tag{2.5}$$

where $\mathbf{y}_i \in \{-1, +1\}$, which are binary responses like spam web pages or non-spam ones. This model is usually used to predict binary responses for classification problems.

Support Vector Machines (SVM): Without diving into the detailed arguments, we briefly introduce a $C-$Support Vector Classification (SVC) model, which is a binary classification model but can be extended into multiple ones, here:

$$\begin{aligned} \min_{\mathbf{w},w_0,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{M}\xi_i \\ \text{subject to} \quad & \mathbf{y}_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, \cdots, M \end{aligned} \tag{2.6}$$

where C is a penalty weight, which is manually tuned, and ξ are slack variables. This form is usually called the primal optimization problem for $C-$SVC. More details about SVC, please refer to Chang et al. [40]. Also, SVM can be adapted to regression problems, please refer to Smola and Schölkopf [184].

The three linear models mentioned above can be formalized into a single optimization framework, shown by Yuan et al. [220]. In particular, we consider the following optimization framework:

$$\min_{\mathbf{w}} \quad \mathcal{R}(\mathbf{w}) + C\sum_{i=1}^{M}\mathcal{L}(\mathbf{w};\mathbf{x}_i,\mathbf{y}_i) \tag{2.7}$$

where $C > 0$ is user-specified for balancing the regularization term $\mathcal{R}$ and the sum of losses. Here, we omit the bias term for the sake of discussion. Three common loss functions are considered:

$$\mathcal{L}_S(\mathbf{w};\mathbf{x}_i,\mathbf{y}_i) = (\mathbf{y}_i - \mathbf{w}^T\mathbf{x}_i)^2 \tag{2.8}$$

$$\mathcal{L}_{LR}(\mathbf{w};\mathbf{x}_i,\mathbf{y}_i) = \log\Big(1 + \exp(-\mathbf{y}_i\mathbf{w}^T\mathbf{x}_i)\Big) \tag{2.9}$$

$$\mathcal{L}_{L2}(\mathbf{w};\mathbf{x}_i,\mathbf{y}_i) = \max(0, 1 - \mathbf{y}_i\mathbf{w}^T\mathbf{x}_i)^2 \tag{2.10}$$

where $\mathcal{L}_S$ corresponds to linear regression, $\mathcal{L}_{LR}$ corresponds to logistic regression and $\mathcal{L}_{L2}$ is usually called L_2 loss SVM. The regularization term $\mathcal{R}$ is used to prevent overfitting and the following L_2 and L_1 regularization terms are commonly used:

$$\mathcal{R}_{L2}(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2 \tag{2.11}$$

$$\mathcal{R}_{L1}(\mathbf{w}) = ||\mathbf{w}||_1 \tag{2.12}$$

All methods here are used in this dissertation in different ways.

2.3 Latent Factor Models

Although a linear model is effective in many prediction tasks, it does not usually discover hidden patterns within datasets and cannot be easily used to reveal relationships between features. Here, we review several significant aspects of latent variable models that are related to this dissertation. An extensive overview of a unified view of latent factor models can be found in Banerjee et al. [18], Singh and Gordon [181] and Singh [180]. In this subsection, we briefly review a unified view treatment of latent factor models introduced in Singh and Gordon [181] and Singh [180]. The arguments presented here stand on the definition of regular exponential family and generalized Bregman divergence.

In this dissertation, we focus on two types of latent factor models: 1) matrix factorization and 2) collective matrix factorization, where a number of methods discussed in later chapters in the dissertation can be categorized into these two categories.

2.3.1 Matrix Factorization

The basic matrix factorization model focused in this dissertation can be written as $\mathbf{X} \approx f(\mathbf{U}\mathbf{V}^T)$. Different choices of the prediction link function f, the defition of $\approx$, and the constraints we place on the factors $\mathbf{U}$ and $\mathbf{V}$ lead to a wide range of latent variable models.

The relationship between matrix factorization and exponential families is made clear by viewing the data matrix as a collection of samples $\{\mathbf{X}_{11}, \cdots, \mathbf{X}_{MN}\}$. Let $\mathbf{\Theta} = \mathbf{U}\mathbf{V}^T$ be the parameters. For a regular Bregman divergence, minimizing $\mathbb{D}_{F^*}(\mathbf{X}_{i,j} \,||\, f(\mathbf{\Theta}_{i,j}))$ is equivalent to maxmimizing the log-likelihood of the data under the assumption that $\mathbf{X}_{i,j}$ is drawn from the distribution in ψ_F with natural parameter $\mathbf{\Theta}_{i,j}$.

Here, we provide a more generic treatment of matrix factorization as follows. A matrix factorization can be defined by choosing the following criteria:

1. Data weights $\mathbf{W} \in \mathbb{R}_+^{M \times N}$.
2. Prediction link $f : \mathbb{R}^{M \times N} \to \mathbb{R}^{M \times N}$.
3. Hard constraints on factors $\mathbf{U}, \mathbf{V} \in \mathcal{C}$.
4. Weighted loss $\mathcal{L}$ between $\mathbf{X}$ and $\hat{\mathbf{X}} = f(\mathbf{U}\mathbf{V}^T)$ where $\mathcal{L}(\mathbf{X} \,||\hat{\mathbf{X}}, \mathbf{W}) \geq 0$.
5. Regularization penalty, $\mathcal{R}(\mathbf{U}, \mathbf{V}) \geq 0$.

Given these choices the optimization for the model $\mathbf{X} \approx f(\mathbf{U}\mathbf{V}^T)$ is:

$$\underset{(\mathbf{U},\mathbf{V}) \in \mathcal{C}}{\arg\min}\, \mathcal{L}(\mathbf{X} \,||f(\mathbf{U}\mathbf{V}^T), \mathbf{W}) + \mathcal{R}(\mathbf{U}, \mathbf{V})$$

Prediction links allow nonlinear relationships between $\mathbf{\Theta} = \mathbf{U}\mathbf{V}^T$ and the data $\mathbf{X}$. We focus on the case where $\mathcal{L}$ is a generalized Bregman divergences defined in Definition 4 and f is the matching link.

A lot of existing popular methods can be formalized under the generic view of matrix factorization. Here, we review several ones that are related to this dissertation:

Singular Value Decomposition (SVD): The domain of $\mathbf{X}_{i,j}$ is set to $\mathbb{R}$ and the link function $f(\boldsymbol{\theta}) = \boldsymbol{\theta}$. The loss $\mathcal{L}$ is chosen to be $||\mathbf{W} \odot (\mathbf{X} - \hat{\mathbf{X}})||_F^2$ while $\mathbf{W}_{i,j} = 1$ for regular SVD and $\mathbf{W}_{i,j} \geq 0$ for weighted SVD. For SVD, it is sometimes to constraint $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^T\mathbf{V} = \mathbf{\Lambda}$ to ensure the unique solution.

Probabilistic Latent Semantic Indexing (pLSI)/ Probabilistic Latent Semantic Analysis (pLSA): The domain of $\mathbf{X}$ follows $\mathbf{1} \circ \mathbf{X} = 1$ and the link function $f(\boldsymbol{\theta}) = \boldsymbol{\theta}$. The loss $\mathcal{L}$ is chosen to be $\sum_{i,j} \mathbf{W}_{i,j}(\mathbf{X}_{i,j} \log \frac{\mathbf{X_{i,j}}}{\hat{\mathbf{X}}_{i,j}})$ while $\mathbf{W}_{i,j} = 1$. The constraints on latent factors are $\mathbf{1}^T\mathbf{U1} = 1$ while $\mathbf{U}_{i,j} \geq 0$. and $\mathbf{1}^T\mathbf{V} = \mathbf{1}$ while $\mathbf{V}_{i,j} \geq 0$.

Non-negative Matrix Factorization (NMF): The domain of $\mathbf{X}$ is set to $\mathbb{R}_+$ and the link function $f(\boldsymbol{\theta}) = \boldsymbol{\theta}$. The loss $\mathcal{L}$ is chosen to be $\sum_{i,j} \mathbf{W}_{i,j}(\mathbf{X}_{i,j} \log \frac{\mathbf{X_{i,j}}}{\hat{\mathbf{X}}_{i,j}} + \hat{\mathbf{X}}_{i,j} - \mathbf{X}_{i,j})$ while $\mathbf{W}_{i,j} = 1$. The constraints on latent factors are $\mathbf{U}_{i,j} \geq 0$ and $\mathbf{V}_{i,j} \geq 0$.

2.3.2 Collective Matrix Factorization

A set of related matrices involves entity types $\mathcal{E}_1, \cdots, \mathcal{E}_t$, where the elements of each type are indexed by a row or column in at least one of the matrices. The number of entities of type $\mathcal{E}_i$ is denoted N_i. The matrices are denoted $\mathbf{X}^{(i,j)}$ where each row corresponds to an element of type $\mathcal{E}_i$ and each column to an element of type $\mathcal{E}_j$. Each data matrix can be factored under the model discussed above, $\mathbf{X}^{(i,j)} \approx f^{(i,j)}(\boldsymbol{\Theta}^{(i,j)})$ where $\boldsymbol{\Theta}^{(i,j)} = \mathbf{U}^{(i)}(\mathbf{U}^{(j)})^T$.

Collective matrix factorization addresses the problem of simultaneously factoring a set of matrices that are related, where the rows or columns of one matrix index the same type as the row or column of another matrix. Note that if a matrix is unrelated to the others, it can be factored independently. Here, we consider the schema $E = \{(i,j) : \mathcal{E}_i \sim \mathcal{E}_j \wedge i < j\}$. We assume that each matrix in the set $\{\mathbf{X}^{(i,j)}_{(i,j)\in E}\}$ is reconstructed under a weighted generalized Bregman divergence with factors $\{\mathbf{U}^i\}_{i=1}^t$ and constant data weight matrices $\{\mathbf{W}^{(i,j)}\}_{(i,j)} \in E$. The total reconstruction loss on all the matrices is the weighted sum of the losses for each reconstruction:

$$\mathcal{L}_u = \sum_{(i,j)\in E} \alpha^{(i,j)} \mathbb{D}_F(\boldsymbol{\Theta}^{(i,j)} \,||\, \mathbf{X}^{(i,j)}, \mathbf{W}^{(i,j)}) + \sum_{i=1}^{t} \mathcal{R}(\mathbf{U}^{(i)}) \tag{2.13}$$

where $\alpha^{(i,j)} \geq 0$ are weights for each individual loss and we regularize on a per-factor basis to mitigate overfitting. The learning process is to find all factors to minimize $\mathcal{L}_u$.

Chapter 3

Information Filtering in CQA Portals

3.1 Introduction

Beginning from this chapter, we focus on the problem of information filtering in online conversational media. We start the discussion on a particular type of such media, Community-based Question Answering (CQA) portals in this chapter and expand the discussion to other media as well in later chapters. Two general approaches to tackle the problem of information filtering will be explored in this chapter: 1) building linear models based on features and 2) identify discriminative or important features from some characteristics of CQA. In fact, the techniques developed in later chapters can be seen as more advanced versions to the methodologies demonstrated in this chapter.

CQA portals are important to online onversational media in many aspects. First of all, content in CQA is usually presented in an informal fashion, carrying out as conversations between users, although these users may not see or know each other at all. Second, CQA plays a significant role in online conversational media. People ask questions and help

others to seek answers on numerous type of CQA portals. For instance, Yahoo! Answers announced[1] in 2010 that the site has served 1 billion answers since the launch of the site in 2004 and more than 0.8 million questions per day. Given this scale, information filtering would be vital for users to obtain relevant information and satisify their information needs.

In this chapter, we tackle the problem of information filtering in CQA portals. In partiuclar, we explore the problem of extracing question answering content from online forums in section 3.2. This functionality is especially useful for the services that are not fully designed for question answering portals, such as discussion boards and forums. We demonstrate that a linear model with simple features can outperform even complex models in the task. In section 3.3, we explore an important feature–users' authority score–and see that it can improve filtering performance in CQA portals.

3.2 Mining Questions and Answers in CQA

Discussion boards, also known as online forums, are popular web applications widely used in different areas including customer support, community development, interactive reporting and online education. Online users share ideas, discuss issues and form communities within discussion boards, generating a large amount of content on a variety of topics. As a result, interest in knowledge discovery and information extraction from such sources has increased in the research community.

While the motivation for users to participate in discussion boards varies, in many cases, people would like to use discussion boards as problem-solving platforms. Users post questions, usually related to some specific problem, and rely on others to provide potential answers. Numerous commercial organizations such as Dell and IBM directly use discussion boards as problem-solving solutions for answering questions and discussing needs posed by customers. Cong et al. [54] found that 90% of 40 discussion boards they

[1] http://yanswersblog.com/index.php/archives/2010/05/03/1-billion-answers-served/

investigated contain question-answering knowledge. Using speech acts analysis on several sampled discussion boards, Kim et al. [114, 113] showed that question answering content is usually the largest type of content on discussion boards in terms of the number of user-generated posts. Therefore, mining such content becomes desirable and valuable.

Mining question answering content from discussion boards has several potential applications. First, search engines can enhance search quality for question or problem related queries by providing answers mined from discussion boards. Second, online Question Answering (QA) services such as *Yahoo! Answers*[2], *Answers.com*[3] and *AllExperts*[4] would benefit from using content extracted from discussion boards as potential solutions or suggestions when users ask questions similar to what people have discussed on forums. This would eliminate the time users wait for answers and enrich the knowledge base of those QA services as well since discussion boards have a longer history than that of QA services and also own a much larger amount of user generated content. Third, users who often provide questions in forums may have expert knowledge in particular areas. Researchers are trying to find experts in social media by utilizing question answering content; authorities are discovered in discussion boards by understanding question answering content and user interactions [30, 223, 110]. In addition, question answering content extracted from discussion boards can be further used to augment the knowledge base of automatic chat-bots [71, 101].

Although general content mining of discussion boards has gained significant attention in recent years, the retrieval of question and potential answers from forums automatically and effectively is still a non-trivial task. Users typically start a thread by creating an initial post with arbitrary content and others reply to it in accordance with the type of the first post. For example, if the first post is about a question, following posts may

[2]http://answers.yahoo.com/
[3]http://www.answers.com/
[4]http://www.allexperts.com/

contain similar experiences and potential solutions. If the first post is an announcement, following posts may contain clarifications, elaborations and acknowledgments. Hence, due to the existence of different types of information, we cannot assume that every thread on a discussion board is about a question, which makes discussion boards fundamentally different from QA services like Yahoo! Answers that are designed specifically for question answering. Additionally, the asynchronous nature of discussion boards makes it possible or even common for multiple users to pursue different questions in parallel within one thread.

In this section, we explore the problem of extracting question answering content from discussion boards and divide it into two subtasks: identifying question-related first posts and finding potential answers in subsequent responses within the corresponding threads. We address both subtasks as classification problems and focus on the following research questions:

- Can we detect question-related threads in an efficient and effective manner? In addition to the content itself, what other features can be used to improve the performance? How much can the combinations of some simple heuristics improve performance?

- Can we effectively discover potential answers without actually analyzing the content of replied posts? Who contributes those posts and where do those posts usually appear?

- Can this task be treated as a traditional information retrieval problem suitable to a relevance-based approach to the retrieval of question-answering content?

We choose several content-based and non-content based features and carefully compare them individually and also in combinations. We do not use any service- or dataset-specific heuristics or features (like the rank of users) in our classification model; therefore our

approach should be usable in any discussion board. In order to test whether our method can improve performance in both subtasks, we mainly compare our approach with one recent similar work [54] (to our knowledge, the first to attack the same problem) and show significant improvements in experimental results.

Sub-section 3.2.1 defines our tasks in more detail. Sub-section 3.2.2 presents our features and gives a simple overview of other approaches from previous work. Experimental results are reported in sub-section 3.2.3. Sub-section 3.2.4 concludes the whole section.

3.2.1 Problem Definition

Here, we discuss the problem in detail and then present a definition of the problem.

Questions: If the first post of one thread is about a specific problem that needs to be solved, we would consider that post as a whole to be a question post. We do not focus on identifying "question sentences" or "question paragraphs" but instead to find whether the first post is a "question post". Since users often express their problems in an informal way and questions are stated in various formats, it is difficult to recognize questions at the sentence or even paragraph level. For example, the following paragraph is a question post from UbuntuForums.org, the official discussion board of Ubuntu Linux:

> *There are a number of threads on Firefox crashes, so it's nothing new. I upgraded from U8.04 to U8.10, but it's no better. Then I tried Seamonkey, and it worked fine for a couple of days. Now it too is crashing. I'm baffled. Anyone have any ideas what I can do?*

Although the last sentence is a question sentence, it gives us little information about what the real problem is. The true problem is the scenario the author described with several sentences as a whole. This post has another paragraph providing machine configurations which we do not include here. Therefore, it is reasonable to treat the whole post as a

question post. If there are multiple questions discussed in the first post, the interaction in following replied posts might become complex (e.g., users may answer all those questions while others may only response to some of them). To simplify the task, we treat it as a single question post.

Answers: If one of the replied posts contains answers to the questions proposed in the first post, we regard that reply as an answer post. As we discussed above, we do not consider the number of answers should match the number of questions. Additionally, we only consider those replies that directly answer the questions from the first post. We ignore other questions (usually elaborated from the original ones) within replied posts and their corresponding answers. Although such answers may provide more information to the original questions and therefore could be potential better answers, in reality, users need to understand all replied posts above to get an overall idea and answers would become less meaningful if we only extract that single reply as the answer to the first post. We also consider replied posts not containing the actual content of answers but providing links to other answers as answer posts. If multiple posts provide links to other potential answers, we treat the first one as the answer post.

Definition 5. *A discussion board is a collection of threads. Each thread consists of the first post and following replied posts. Our task is:*

1. *To detect whether the first post is a "question post" containing at least one problem needed to be solved.*
2. *If the first post is a "question post", try to identify the best answer post either directly answering at least one question proposed in the first post or pointing to other potential answer sources.*

Therefore, the result from our system is question-answer post pairs. Ideally, users do not need other information (e.g., the posts between them) to understand these pairs.

3.2.2 Classification Methods

We consider both subtasks described in Section 3.2.1 as classification problems. Here, we introduce the features we use and a brief review of previous approaches.

Question Detection: For this subtask, we describe and use several features other researchers have used previously (e.g., question mark, 5W1H words) as well as features that are borrowed from other fields (e.g., N-gram).

- Question mark: If users want to ask a question, they may express it in a question sentence and therefore the sentence may contain a question mark at the end.

- 5W1H Words: If there is a question sentence, users probably would use 5W1H words in it.

- Total number of posts within one thread: From our empirical study we found that if one thread has many posts, either the topic of the thread probably shifts or the original first post may not contain enough information and hence further clarifications or elaborations are needed. Both cases are not in our problem definition.

- Authorship: Who would usually ask questions? Recent work shows that high quality content is generated by highly authoritative authors in social media (e.g., Agichtein et al. [4] and Hu et al. [100]). In our context, we consider high quality contents to be answers and highly authoritative authors are users who usually answer others' questions. Therefore, by contrast, fresh users are more likely to post questions rather than answering questions and a large portion of total posts (including all replies) a fresh user makes are likely all questions.

- N-gram: Carvalho and Cohen [37] suggested that n-grams would improve speech acts analysis on E-mail. The task is similar to our work and therefore we would like to see whether this feature works for discussion boards.

In summary, we use the number of question marks, the number of each 5W1H words, total number of posts within one thread and authorship (the number of posts one user starts and the number of posts one user replies) as features.

Answer Detection: In this subtask, we focus on how to detect answer posts without analyzing the content of each post using natural language processing techniques. We are also interested in how non-content features can contribute to classification results.

- The position of the answer post: According to our definition of the problem, we notice that the answer post usually appears not very close to the bottom if the question receives a lot of replies.
- Authorship: Same as the last subtask.
- N-gram: Same as the last subtask.
- Stop words: Although "stop words" are usually regarded as "noise words", we want to see whether the author of answer posts would use more detailed and precise words rather than "stop words", in contrast to other types of posts such as elaborations, suggestions and acknowledgement.
- Query Likelihood Model Score (Language Model): We use this basic language model method to calculate the likelihood that a replied post is relevant to the original question post. We use this feature as an example to show how a relevance-based model performs in the task.

In summary, we use the position of the answer post, the authorship, N-gram, the count of each stop word and the score of Query Likelihood Model as features.

Other methods: We principally compare our method with the approaches introduced by Cong et al. [54], a recent work addressing a similar problem. To detect the questions,

they used the supervised learning approach Sequential Pattern Mining. First, each sentence is preprocessed by a POS tagger only leaving 5W1H words and modal words. Then the sequential patterns are generated by a modified version of the PrefixSpan algorithm [156] to incorporate both minimum support and minimum confidence, which are assigned empirically. They treat all generated patterns as features. They considered "finding answers" as a retrieval problem. The retrieval model they introduced is a graph-based model incorporating inter-post relevance, authorship and the similarity between replied posts and the first post. They showed two variations of the graph-based model; one that is combined with the Query Likelihood language model and another combined with the KL-divergence language model. We implement all these methods and compare them in our experiments. Notice that they did not explicitly define what "question" or "answer" is. Therefore, our task may be slightly different from theirs.

3.2.3 Experiments

We selected two discussion boards as our data sources. We crawled 721,442 threads from Photography On The Net[5], a digital camera forum (DC dataset), and 555,954 threads from UbuntuForums[6], an Ubuntu Linux community forum (Ubuntu dataset).

For the question detection subtask, we randomly sampled 572 threads from the Ubuntu dataset and 500 threads from the DC dataset. We manually labeled all first posts in these threads into question posts and non-question posts using our criteria introduced in Section 3. For the answer detection subtask, we selected 500 additional question-related threads from both data sources. Therefore, we have 2,580 posts in total (including the first posts) from the Ubuntu dataset and 3,962 posts in total (including the first posts) from the DC dataset. We manually labeled all posts into answers and non-answers. We note that in accordance with our problem definition, only one answer post per thread is labeled as such

[5]http://photography-on-the.net/
[6]http://ubuntuforums.org/

(the remainder are labeled as non-answers).

We preprocessed all posts by modifying possible abbreviations into their full form (e.g., "we're" into "we are", "it's" into "it is") and stemming all words. For Sequential Pattern Mining, the Stanford Log-linear Part-Of-Speech Tagger [188] was used and minimum support and minimum confidence were set to 1.5% and 80% respectively. For N-grams, we generated 3,114 N-grams (1-5 grams) from the Ubuntu dataset and 1,604 N-grams from DC dataset for question detection while 2,600 N-grams from Ubuntu dataset and 1,503 N-grams from DC dataset for answer detection. For stopwords, we used 571 normal stop words.[7] We use LIBSVM 2.88 [39] as our classifier and all classification results are obtained through 10-fold cross validation. In order to avoid classification bias and get better results, we balanced our data into around 50% positive samples versus 50% negative samples in all experiments. For example, we have 500 positive instances and 2080 negative instances for answer detection on Ubuntu dataset. Therefore, we replicated the positive training instances four times to give 2,000 examples (but left the test set unchanged). Since in any real settings, the data is inherently skewed, a better learning approach such as cost-sensitive learning may be more realistic. Table 3.1 shows all the features we used and their abbreviations.

Question Detection: We first evaluate the performance of features introduced in subsection 3.2.2 individually. Table 3.3 gives the results of precision, recall, F-measure and accuracy (sorted by accuracy) of the Ubuntu dataset and Table 3.4 shows the results from the DC dataset. It is easily to notice that *Length*, *5W1H* and *Question Mark*, three simple heuristics, generally cannot give good performance while *Sequential Pattern Mining* always outperforms these simple methods on both datasets, which validates the experiments performed by Cong et al. [54]. Additionally, the results show that *Authorship* is a much better heuristic and can achieve reasonable performance compared with *Sequential Pattern*

[7] http://www.lextek.com/manuals/onix/stopwords2.html

Table 3.1: The Features and Their Abbreviations

Features	**Abbreviation**
Question Mark	QM
5W1H Words	5W
Total # Posts	LEN
Sequential Patterns	SPM
N-grams	NG
Authorship	AUTH
Position	POSI
Query Likelihood Model	LM
Stop Words	SW
Graph+Query Likelihood Model	GQL
Graph+KL-divergence Model	GKL

Table 3.2: Example N-grams from DC Question Dataset

i do not know if	i wa wonder if anyon
what is the best way	i do not have
i am not sure	do not know what
i am look for	i can not
do not know	would like to

Mining although it seems that performance may be highly dataset dependent. On both dataset, *N-grams* achieves the best performance in all metrics in terms of a single type of feature. This phenomenon suggests that users do use certain language patterns to express problems and questions in discussion boards. Table 3.2 shows 10 sample N-grams extracted from DC dataset that used for question detection. Note that the results are stemmed words. Since *N-grams* and *Sequential Pattern Mining* (which requires a POS tagger) are relatively complicated methods (vs. simple heuristics such as finding question marks and 5W1H words), the computational effort may be impractical for large datasets. In order to avoid high computation methods, we do further experiments on the combinations of those simple methods and see whether the performance can be improved and therefore we can use simple combinations as alternatives.

Table 3.3: Single Feature Ubuntu Question

Features	Prec.	Recall	F1	Accu.
LEN	0.568	0.936	0.707	0.623
5W	0.613	0.759	0.679	0.651
QM	0.649	0.634	0.641	0.656
AUTH	0.700	0.725	0.712	0.716
SPM	0.692	0.829	0.754	0.738
NG	0.770	0.906	0.833	0.823

Table 3.4: Single Feature DC Question

Features	Prec.	Recall	F1	Accu.
5W	0.601	0.429	0.500	0.579
LEN	0.564	0.730	0.636	0.590
QM	0.578	0.779	0.664	0.612
SPM	0.642	0.702	0.671	0.661
AUTH	0.723	0.791	0.755	0.748
NG	0.752	0.799	0.775	0.772

Table 3.5 and Table 3.6 show the combinations of simple features compared to *N-grams* and *Sequential Pattern Mining*. We observe that the performance can be improved by combining features. Specifically, *Authorship+Question Mark+5W1H Words+Length* achieved similar or even better results than *Sequential Pattern Mining* on both datasets. Notice that the computation of these features is much simpler than *Sequential Pattern Mining*. In addition, *Question Mark+5W1H Words+Length*, which only require local information, also achieved reasonable performance compared to those feature individually since *Authorship* needs global information. From these results, we found that although these features individually cannot give much evidence reflecting whether a post concerns a question, the combination of them is able to characterize the first post and interestingly none of these simple features attempts to understand the real semantics of the question posts.

Answer Detection: For this subtask, we first did the experiments using individual

Table 3.5: Combined Features Ubuntu Question

Method	Prec.	Recall	F1	Accu
QM+LEN	0.657	0.655	0.656	0.666
AUTH+LEN	0.679	0.757	0.716	0.708
5W+LEN	0.673	0.821	0.740	0.719
QM+5W	0.756	0.636	0.691	0.723
QM+5W+LEN	0.744	0.701	0.722	0.738
SPM	0.692	0.829	0.754	0.738
AUTH+QM+5W+LEN	0.731	0.762	0.746	0.748
NG	0.770	0.906	0.833	0.823

Table 3.6: Combined Features DC Question

Method	Prec.	Recall	F1	Accu.
QM+5W	0.614	0.764	0.681	0.648
5W+LEN	0.627	0.709	0.666	0.650
SPM	0.642	0.702	0.671	0.661
QM+LEN	0.656	0.764	0.706	0.687
QM+5W+LEN	0.672	0.755	0.711	0.698
NG	0.752	0.799	0.775	0.772
AUTH+LEN	0.813	0.874	0.843	0.839
AUTH+QM+5W+LEN	0.863	0.889	0.876	0.876

features, as we did in Question Detection. In order to compare with the methods introduced by Cong et al. [54], we used the ranking score from their retrieval models as a feature to train our classifier. Since *Graph-based model+Query Likelihood Model* and *Graph-based model+KL-divergence Model* performs similarly on both datasets, we only use *Graph-based model+Query Likelihood Model* in this subtask as an example. Table 3.7 and Table 3.8 show the experimental results. In general, *Language Model* and *Graph+Query Likelihood Model* did not perform well using the ranking score as features. The possible reason is that these methods are mainly based on relevance retrieval models, which aim to find the information most relevant to the query (in our case, the question posts). Since all posts within a question thread may be more or less relevant to the question, it is difficult to

Table 3.7: Single Feature Ubuntu Answer

Method	Prec.	Recall	F1	Accu.
GQL	0.673	0.575	0.620	0.650
Stopword	0.665	0.617	0.640	0.655
NG	0.690	0.638	0.663	0.678
LM	0.717	0.650	0.682	0.699
POSI	0.743	0.730	0.737	0.712
AUTH	0.715	0.823	0.765	0.721

Table 3.8: Single Feature DC Answer

Method	Prec.	Recall	F1	Accu.
GQL	0.661	0.535	0.591	0.628
LM	0.726	0.603	0.659	0.685
AUTH	0.680	0.800	0.735	0.710
NG	0.735	0.680	0.706	0.716
Stopword	0.730	0.696	0.712	0.717
POSI	0.780	0.88 0	0.827	0.815

rank them and distinguish the best answers from others based on content relevance or similarity measurement. In addition, relevance-based models may unable to handle big lexical gaps between questions and answers. We show one example from UbuntuForums below:

Table 3.9: Combined Features Ubuntu Answer

Method	Prec.	Recall	F1	Accu.
LM+GQL	0.726	0.718	0.722	0.695
Stopword+NG	0.735	0.786	0.760	0.726
LM+POSI	0.733	0.812	0.770	0.733
LM+Stopword	0.758	0.764	0.761	0.735
LM+AUTH	0.739	0.840	0.786	0.748
POS+Stopword	0.785	0.811	0.798	0.773
LM+POSI+Stopword	0.785	0.814	0.799	0.774
LM+POSI+AUTH	0.929	0.964	0.946	0.940
POSI+AUTH	0.935	0.969	0.952	0.946

The first post:

> *can any one help me load ubuntu 8.10 on to my pc? i have a asus AS V3-P5V900 but when i load from cd it keeps crashing , i think i dose not reconise the graphics card. when i boot from cd it asks me what lauguge ENGLISH then when try to load it crash again i have tryed help and put in via=771 any help please ?*

The answer post:

> *You might try using the "Alternate" install CD:*
> *http://www.ubuntu.com/getubuntu/downloadmirrors#alternate*

Notice that this answer post contains a web link while all "keywords" (e.g., ubuntu 8.10, asus AS V3-P5V900, crash and etc.) in the first post do not appear in the answer post. If we calculate Query Likelihood Model score for the answer post, nearly all words in the question post can only receive "background" smoothing score and hence the model would rank this post "irrelevant". Essentially the same situation happens when using similarity measurement (e.g., cosine similarity). *N-gram* did not outperform other features in this subtask, which suffers from various expressions in answer posts. Interestingly, the *Stopword* approach has performance similar to *N-gram* in both datasets. *N-gram* usually requires more computational effort than *Stopword* since *Stopword* has a fixed number of features for all datasets while *N-gram* needs to be generated separately and usually contains thousands of features. Therefore, in our later experiments, we use *Stopword* instead of *N-gram*. We also note that *Authorship* and *Position*, two simple heuristics, perform reasonably well and achieve comparatively high F1-Score on both datasets.

Inspired by question detection subtask, we conducted experiments using combinations of features on the two datasets. Tables 3.9 and 3.10 provide the corresponding results.

Table 3.10: Combined Features DC Answer

Method	Prec.	Recall	F1	Accu.
LM+GQL	0.735	0.594	0.657	0.688
LM+AUTH	0.700	0.771	0.734	0.719
Stopword+NG	0.737	0.688	0.712	0.720
LM+Stopword	0.765	0.717	0.740	0.747
LM+POSI	0.780	0.879	0.827	0.815
LM+POSI+Stopword	0.846	0.899	0.872	0.867
POSI+Stopword	0.846	0.901	0.873	0.868
LM+POSI+AUTH	0.951	0.991	0.970	0.970
POSI+AUTH	0.958	0.993	0.975	0.975

In this subtask, we not only combine simple heuristics but also combine non-content features and content-based features. The first interesting finding is that *Position+Authorship* outperforms all other feature combinations and greatly improves the performance. This would explain that senior members usually answer questions in certain positions (e.g., near to the top post). This combination is easy to compute and there are no other parameters to tune. In order to better understand how these two features contribute to the final results, we plot them in Figure 3.1 and Figure 3.2 for both datasets. The X-axis shows the ratio of the number of starting posts versus follow-up posts for users who answered questions in our datasets. The Y-axis shows the ratio of the position of answer posts from the top of the thread versus to the bottom. Both figures demonstrate the obvious signal that most answer posts are close to the top when the author of these posts are senior users who usually write replies rather than starting posts.

We also notice that the combination of content-based features (e.g., *Language Model*, *Stop words*) and non-content features (e.g., *Position*, *Authorship*) may also get better results compared to Table 3.9 and Table 3.10. The *Position+Stopword* combination performed reasonably well on both datasets, only requires local information, and is simpler

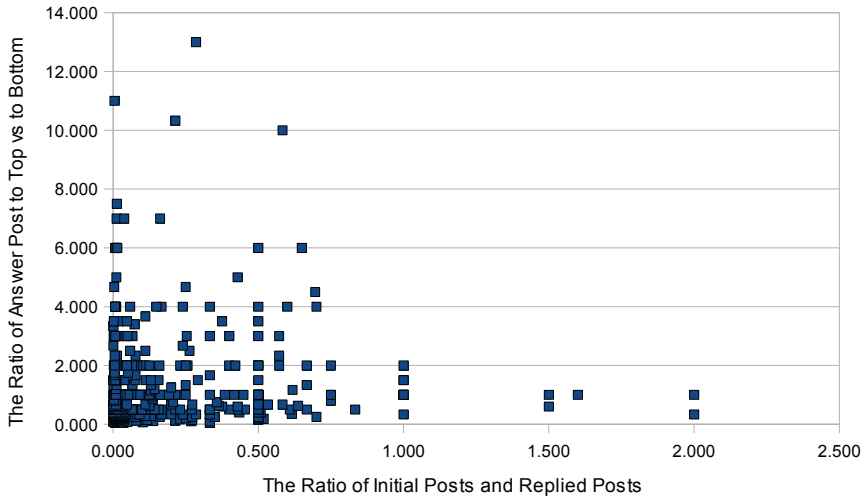

Figure 3.1: Authorship and Position on Ubuntu

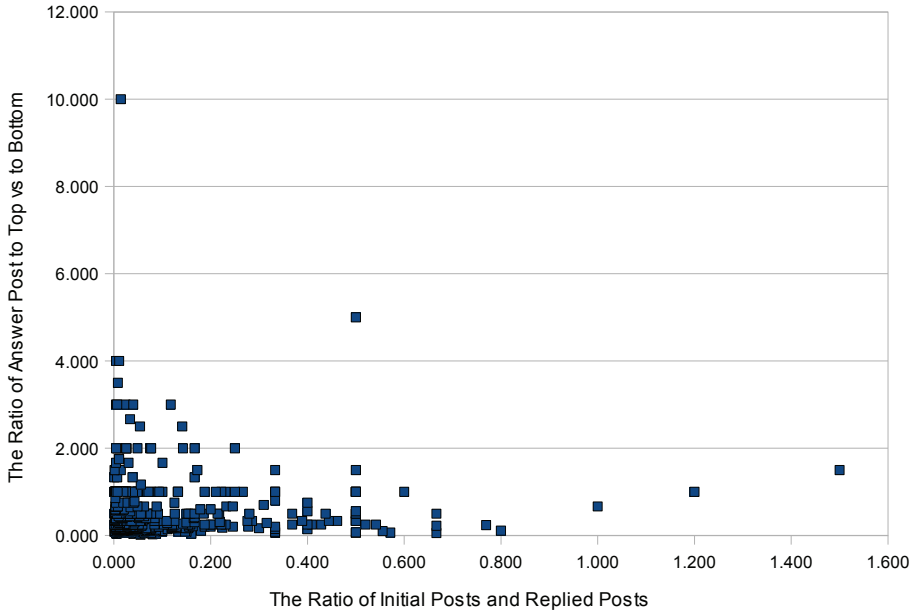

Figure 3.2: Authorship and Position on DC

Table 3.11: Ranking Scheme

	Ubuntu		**DC**	
Method	P@1	MRR	P@1	MRR
LM	0.352	0.559	0.274	0.468
GQL[54]	0.360	0.570	0.220	0.414
GKL[54]	0.358	0.556	0.223	0.415
POSI+AUTH	0.902	0.949	0.928	0.964

than any kind of relevance-based features. In general, we can see that performance benefits from a combination of features, especially those simple features. Additionally, the combination of non-content and content features also improves performance significantly.

Other Experiments: We also propose a simple ranking scheme based on the classification method. The ranking score is simply computed by linearly combining position and authorship information:

$$s = \alpha * V_1 + (1 - \alpha) * \beta * V_2 + (1 - \alpha) * (1 - \beta) * V_3$$

where V_1,V_2 and V_3 are scores from classifiers of combination of position and authorship, position only and authorship only respectively. α and β are empirical parameters and we set 0.6 to both of them. Table 3.11 shows the results compared to basic Query Likelihood Language Model, Graph-based+KL-divergence model proposed by [54] in terms of Precision@1 and Mean Reciprocal Rank (MRR) where MRR is the mean of the reciprocal ranks of the answers over a set of questions. Our ranking scheme outperforms other previous relevance-based approaches.

3.2.4 Summary

In this section we defined the problem of selecting Question and Answer post pairs from discussion boards and addressed it as a classification problem. The contributions include:

1. We show that the use of N-grams and the combination of several non-content features can improve the performance of detecting question-related threads in discussion boards.

2. We show that the number of posts a user starts and the number of replies produced and their positions are two crucial factors in determining potential answers.

3. We show that relevance-based retrieval methods would not be effective in tackling the problem of finding possible answers but the performance can be improved by combining with non-content features while we treat retrieval scores as features.

4. Using classification results, we are able to design a simple ranking scheme that outperforms previous approaches when retrieving potential answers from discussion boards.

This section explicitly defines the problem of selecting question answering post pairs from discussion boards and shows better performance compared to previous approaches. We believe that this is a first step toward a better understanding of the interaction of question answering in such media.

3.3 Mining Participant Reputation in CQA

In the previous section, we explored a number of content features that can improve the performance of a certain information filtering task in CQA. One aspect left is how users' reputation impact the task. In this section, we discuss this direction.

Community-driven Question Answering (CQA) has existed for decades as part of bulletin board systems and Usenet, but has recently been popularized within web portals in which users answer questions posed by other users. CQA has proven to be more effective since users can post natural language questions rather than issuing several word queries to

search engines. One typical example of a CQA system is Yahoo! Answers, which already attracts tens of millions of users and stores hundreds of millions of questions and answers [129]. Unfortunately, users may post similar or identical questions multiple times and the quality of answers varies drastically. Recent work [24] shows that a large portion of content in CQA is not useful for users. On one hand, it is not appropriate for users to re-post existing questions. On the other hand, users may find it difficult to browse within the large question-answer archive. Therefore, there is of increasing interest to build retrieval mechanisms to automatically search in a question-answer archive and provide high quality content to users.

Not surprisingly, much research work (e.g., [4, 93]) has shown that reputation of users are good indicators of the quality and reliability of the content. Many ranking schemes which take advantage of user reputation (e.g., [223, 110, 4]) have been proposed to provide search results to users. The assumption behind these methods is that highly authoritative users may provide high quality content. Since the naturally bipartite structure of CQA where users who post questions and users who provide answers can be seen as two subcommunities within CQA, several ranking approaches (e.g., [223, 110]) derived from the HITS algorithm [118] have been shown to improve retrieval performance. However, there is no evidence to show whether this is the most effective way to model users' expertise. In addition, PageRank-like ranking schemes are less often used to model reputation in a CQA context. One possible reason is that it is relatively difficult to see whether CQA has the "hierarchical ranking structure" that PageRank provides where the reputations of users depend not only on the number of questions and answers a participant produces but also on with whom the user interacts, compared to naturally bipartite structure of HITS.

In this section, we discuss how to use PageRank to model user reputation in CQA. We view the link between users as reflecting the likelihood of one user providing an answer to the other. In addition, we introduce topical link analysis [152], which has shown success

in modeling web page authority, into CQA and show how to incorporate topical information. Our specific contributions include: 1) The use and justification of a PageRank-based method for user reputation modeling in CQA; 2) The introduction of topical link analysis for user reputation modeling in CQA. The method does not use any site-specific features and can be easily applied to other social media; 3) Showing how probabilistic Latent Semantic Analysis (pLSA) can be embedded into user reputation modeling; 4) A comparative study of several popular user reputation modeling schemes in terms of retrieving best answers in Community Question Answering services.

In subsection 3.3.1, we review several existing user reputation modeling schemes and discuss how to use PageRank in CQA. In subsection 3.3.2, we discuss topical link analysis in CQA and its challenges. Subsection 3.3.3 describes experimental results showing the effectiveness of different ranking schemes. Subsection 3.3.4 provides discussions and future work.

3.3.1 User Reputation Model Review

We first review several user reputation models based on link analysis and simple heuristics.

HITS-like scheme: Kleinberg [118] identifies two important properties for a web page: hubness and authority, and proposes a mechanism to calculate them effectively. The basic idea behind HITS is that pages functioning as good hubs will have hyperlinks pointing to good authority pages, and good authorities are pages to which many good hubs point. Authority and hub scores of a web page can be computed via mutual reinforcement, which can be described as follows:

$$A(i) = \sum_{j:j\to i} H(j) \tag{3.1}$$

$$H(i) = \sum_{i:i\to j} A(j) \tag{3.2}$$

If we treat each user as a node and if user i answers a question posted by user j, there will be a link pointing from i to j. Therefore, a user who often posts good questions which receive informative answers will have many in-links and a user who often answers questions from others will have many out-links, indicating that the first type of users displays the hubness property and the second type shows authority. Using Equations 3.1 and 3.2, we can calculate hub and authority scores for each user in CQA. Sometimes, however, we need a single score to represent the rank of a user. For example, when combining other models (e.g., relevance models), a single user rank score may help us simplify our overall model. One easy way is to combine them in a linear fashion, which is used in our experiments:

$$UserRank(i) = \gamma * A(i) + (1 - \gamma) * H(i)$$

where γ is a manually tuned balancing parameter to control the relative importance between authority scores and hubs scores.

PageRank-like scheme: Page et al.'s PageRank [153] is a static ranking of web pages based on the measure of prestige in social networks. PageRank can be seen as a random surfer model in which a surfer on a given page i can choose with probability $(1 - d)$ to select uniformly one of its outlinks and with probability d to jump to a random page from the web. The PageRank score of node i is defined as the stationary probability of finding the random surfer at node i. One formulation of PageRank is:

$$PR(i) = (1 - d) \sum_{j:j \to i} \frac{PR(j)}{O(j)} + d\frac{1}{N} \tag{3.3}$$

PageRank is not a popular choice to model user reputation in the context of CQA. One possible reason is that there is no obvious evidence implying that ranking with hierarchical structures is better than the bipartite structure used in HITS (or even as effective). In

addition, Equation 3.3 indicates that a node would share its PageRank score by uniformly distributing the value to each out-going link. However, if we treat each user as a node and there would be a link from user i to j if j answers a question posted by i, it does not make much sense that user i would share its importance to user j since user j should have higher expertise because of answering questions. Furthermore, PageRank needs to randomly "jump" to any page on the Web even when there is no hyperlink between them. This lacks an intuitive explanation since it is difficult to think about a user who can share authority with other users with whom the user never interacts.

In this chapter, we think about the links between user nodes as the ***possibility*** that interactions could happen between users. If no interactions ever happen between two users, they still might invoke interactions in the future with a certain low probability, captured by the "random jump" part of PageRank. If they already have interactions, the probability of their future interactions would be higher than random and indicated by the number of existing interactions, which is captured by "out-going links" part of PageRank. Therefore, the PageRank score of a user measures the activeness of this user.

Other Heuristics: Zhang et al. [223] proposed a heuristic ranking scheme called *Z-Score* that is based on the number of questions and answers one user generates. Z-Score is calculated as follows:

$$Z = \frac{a - n/2}{\sqrt{n}/2} = \frac{a - q}{\sqrt{a + q}}$$

where a is the number of answers provided by one user, q is the number of questions produced by one user, and n is the sum of a and q. Zhang et al.'s rationale for the heuristic is to measure the difference in behavior from a "random" user who posts answers with probability $p = 0.5$ and posts new questions with probability $1 - p = 0.5$. If the user equally asks or answers questions, the z-score will be close to 0. A positive z-score

captures a user who asks answers more than asks. Another simple heuristic is derived from Hong and Davison [93] in which the authors found that the number of posts a user generates and the number of replies a user provides are two good indicators for the user reputation in forums. Here, we use the linear combination of the number of questions and answers a user generates as the model of reputation:

$$SimpleRank = \theta * a + (1 - \theta) * q$$

where a is the number of answers one user provides and q is the number of questions that user produces. The parameter θ is used to control whether we emphasize the capability to post new questions or to answer questions for a user. In our experiment, we use $\theta = 0.8$ to focus on the capability to answer questions.

3.3.2 Topical Link Analysis for User Reputation

So far, all user reputation models we reviewed are trying to give a "global" user reputation score, which means that the score represents the user's authority across all topics. However, one may argue that an expert in Computer & Internet may not give good suggestions in Gardening. Obviously, it is better to give authority scores according to different topics and rank user reputations differently. That is why some ranking schemes are designed to take topical information into account, such as Topic-Sensitive PageRank [83]. Here, we review Topical PageRank [152], one successful topical ranking scheme, and discuss how to adapt it into the context of user reputation modeling in CQA.

Topical PageRank: The main motivation of Topical PageRank is that the authority score of a web page should be modeled with respect to different topics. The basic idea of Topical PageRank is to incorporate topic distribution into the representation of each web page as well as the importance score of each page. Therefore, two vectors are associated

with each page: the content vector and the authority vector. The content vector $\boldsymbol{c}_u$ is a T-dimensional distribution over topics, representing the content of page u, solely determined by content itself. The authority vector $\boldsymbol{a}_u \in \mathbb{R}^T$, a distribution over the same set of topics, is used to measure the importance of the page where $\boldsymbol{a}_{u_k}$ is the importance score on topic k.

Topical PageRank is also a random surfer model. On each page, the surfer may either follow the outgoing links of the page with probability $1 - d$ or jump to a random page with probability d. When following links, the surfer may either stay on the same topic to maintain topic continuity with probability α or jump to any topic i on the target page with probability $1-\alpha$. The probability of jumping to topic k is determined by $\boldsymbol{c}_{u_k}$. When jumping to a random page, the surfer is always assumed to jump to a random topic k. Therefore, the authority score (i) on page u is calculated as follows:

$$\boldsymbol{a}_{u_i} = (1-d) \sum_{v:v\to u} \frac{\alpha \boldsymbol{a}_{v_i} + (1-\alpha)\boldsymbol{c}_{v_i}\boldsymbol{a}_v}{O(v)} + \frac{d}{N}\boldsymbol{c}_{u_i}$$

where $\boldsymbol{a}_v = \sum \boldsymbol{a}_{v_i}$. Note that Nie et al. [152] also proposed a topical version of the `HITS` algorithm, which may be interesting to adapt into CQA in future work.

Adapting Topical PageRank to CQA: One question for Topical PageRank is how to obtain the content vector $\boldsymbol{c}_u$. In [152], a text classifier trained on the pages selected from the twelve top categories (e.g., Arts, Computers, Games) of the dmoz Open Directory Project (ODP) was used. For CQA, a fine-grained topic distribution like Software and Hardware is needed, which is usually hard to obtain. In order to adapt Topical PageRank for CQA, we propose to use an unsupervised learning algorithm to obtain a content vector. In this work, we use probabilistic Latent Semantic Analysis (pLSA) [87], a simple unsupervised topic modeling method. pLSA is a generative model in which documents are not "hard" classified to topics but characterized by a mixture of topics with weights.

The model is to maximize the log-likelihood function

$$L = \sum_{d} \sum_{w} n(d, w) \log P(d) \sum_{z} P(w \mid z) P(z \mid d)$$

where $n(d, w)$ denotes the number of times w occurred in d. The standard computation procedure for maximum likelihood estimation in latent variable models is the Expectation Maximization (EM) algorithm. We do not include details of EM here and readers who are interested can refer to the tutorial in Hong [89].

After knowing the topic distribution of each document (here, in CQA, each question and each answer can be seen as one document), we want to know the topic distribution of each user if we treat users as nodes. One simple way is to add the topic distribution of each document one user U_i generates together. Therefore,

$$P(z|U_i) = \sum_{d \in Q(U_i)} P(z|d)$$

where $Q(U_i)$ represents all the documents user i produces.

Another approach is to introduce a new variable u into the pLSA model to represent users. Therefore, the log-likelihood function is :

$$L = \sum_{u} \sum_{d} \sum_{w} n(d, w, u) \log P(d) \sum_{z} P(u \mid z) P(w \mid z) P(d \mid z)$$

The advantage of this approach is that we can directly obtain the topic distribution for each user through the EM algorithm. However, this would require more computation, especially for large-scale data. In this work, we do not introduce the new variable and focus instead on the simpler method.

3.3.3 Experimental Method

We will compare several ranking schemes, including what we introduced in previous sections. We crawled 59,936 posts from Yahoo! Answers through its API, where 14,122 are questions and 45,814 are answers. Among all answers, 12,731 are selected as "best answers" by users to particular questions. 37,940 unique users are extracted from the dataset. Since we do not have real queries that users issued in Yahoo! Answers for searching the question answer archive, we treat each question in our dataset as a query and all answers as potential answers to the query. Therefore, we have 12,731 questions as queries and their corresponding best answers as relevant results. (We do not consider those questions that have no best answers.) We want to measure ranking schemes in two ways. First, we want to see whether a ranking scheme can return the best answer early in the return-list. Second, we want to see whether the ranking scheme can return more best answers higher than other answers. Specifically, we are looking at these metrics:

- Precision@1(Strict): If the question has the answer selected as the best answer, this best answer should return at the first place if we use the whole question as a query.
- Mean Reciprocal Rank (MRR): Same as the metric above, we want to see the position where the best answer ranked if the question has the best answer chosen by the user.

The above two metrics are strict metrics since each question (or query) only has one answer (or relevant result). In order to evaluate ranking schemes for our second goal, we relax the constraint of returned answers by treating all best answer as relevant results.

- Precision@1(Relaxed): We only want to see whether the top result is a best answer regardless whether it is the best answer selected for the query or not.
- Precision@10: We want to see how many results in top 10 positions are best answers.

Table 3.12: Results of BM25

P@1(S)	MRR	P@1(R)	P@10	MAP
0.0857	0.1414	0.3410	0.3170	0.3081

- Mean Average Precision (MAP): We sum the precision score whenever a new best answer is retrieved and average all scores across all queries (questions).

Since we need to calculate the relevance score for each answer, user reputation model itself is not enough. We combine user reputation model with the Okapi BM25 weighting function. For each answer, we calculate two rankings, one from BM25 and the other one from a user reputation model and combine them with a simple weighted sum:

$$\lambda * rank_{BM25}(a) + (1 - \lambda) * rank_{USER}(a)$$

Results: Unsurprisingly, the parameter λ affects the final results. Thus, for each method and metric, we show how this parameter influences the ranking results. The results of only using BM25 are shown in Table 3.12, where P@1(S) indicates Precision@1(Strict) and P@1(R) indicates Precision@1(Relaxed). We do not include the results of Z-Score in the following discussions since it performs worst in our experiments and the values for each evaluation metric is low. Figure 3.3 and 3.4 show the result of using "Strict Metrics" as λ is varied from 0.8 to 0.9. Two obvious observations can be quickly obtained. First, all the results are worse than using the BM25 ranking result alone. The reason may be that sometimes the best answer could be produced by users that might not be the first authoritative user (e.g., may be second or third). Since "Strict Metrics" only measure whether **the** best answer can be retrieved or not, we argue that the results may not reflect what would happen in real world where users often issue short queries that are less likely to match a whole question. However, this result does give us hints

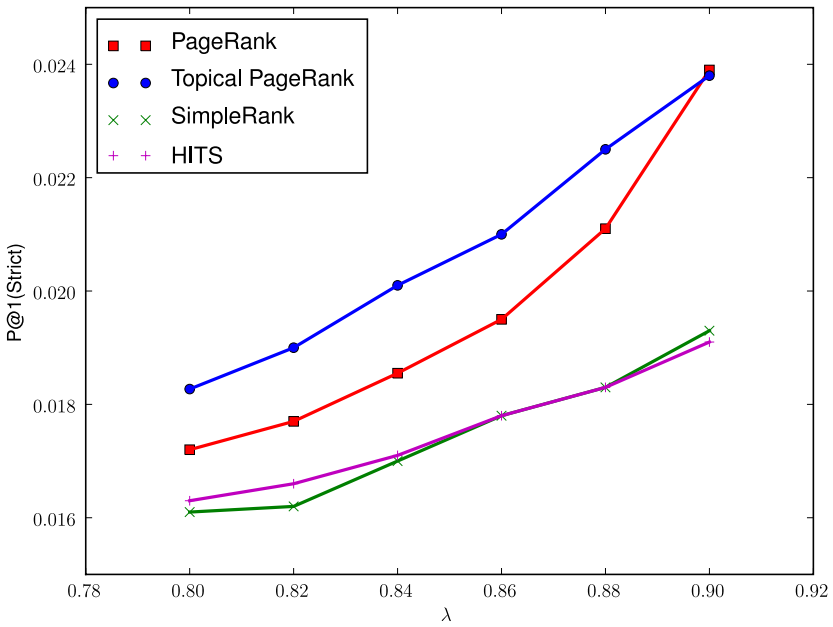

Figure 3.3: P@1 (Strict)

about how different ranking schemes perform in terms of "Strict Metrics", which leads to the second observation that PageRank-like approaches perform better than HITS-like schemes and other heuristics. HITS-like schemes cannot capture the notion of "hierarchical authorities", which means that the user who can answer a question posted by an authority should have higher authority score. PageRank-like approaches naturally model this notion and give better approximation than HITS.

Relaxed Metrics: If we use "Relaxed Metrics", Figures 3.5, 3.6 and 3.7 show that all ranking methods combined with BM25 can improve retrieval performance significantly, which validate the conclusions from other related work that user reputation models can help retrieval tasks in CQA. PageRank-like approaches still outperform simple heuristics and the HITS-like scheme. The results also indicate that SimpleRank performs similarly

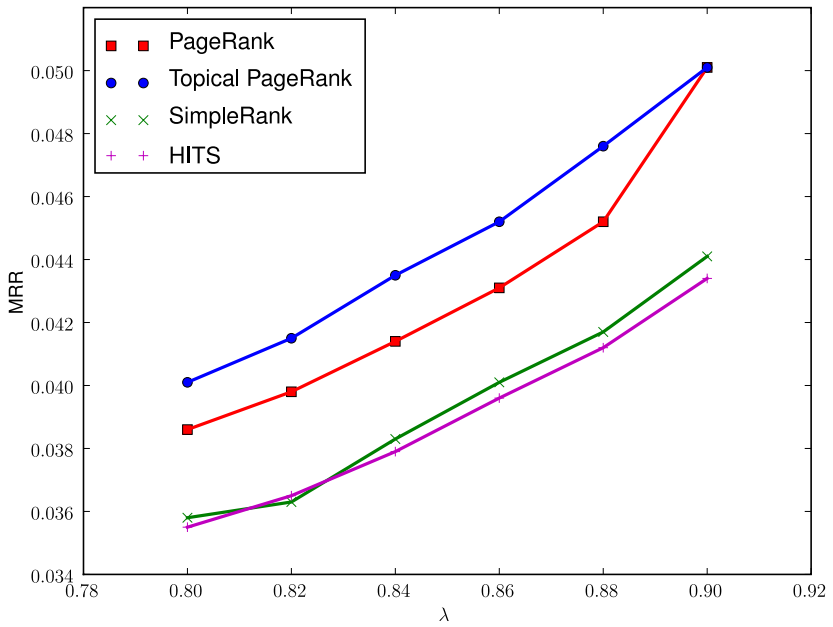

Figure 3.4: MRR

to the HITS-like scheme. This is more evidence that HITS cannot model "hierarchical authorities" as discussed above. One interesting observation is that as λ increases, P@1(Strict), P@1(Relaxed) and MRR also increase but P@10 and MAP decrease. Because the first three metrics only focus on one answer (or relevance result) per question (or query), as we discussed before, user reputation modeling may not help much and P@1(Strict) and MRR are actually worse than only using BM25. On the other hand, if we care about returning more relevant results, P@10 and MAP show the value of user reputation modeling and indicate a significant improvement.

Topical PageRank: For Topical PageRank, we use pLSA as our topic model and specify 20 latent variables (topics). Since our dataset is from the Computer & Internet category of Yahoo! Answers, which has 7 sub-categories, we arguably think the number of

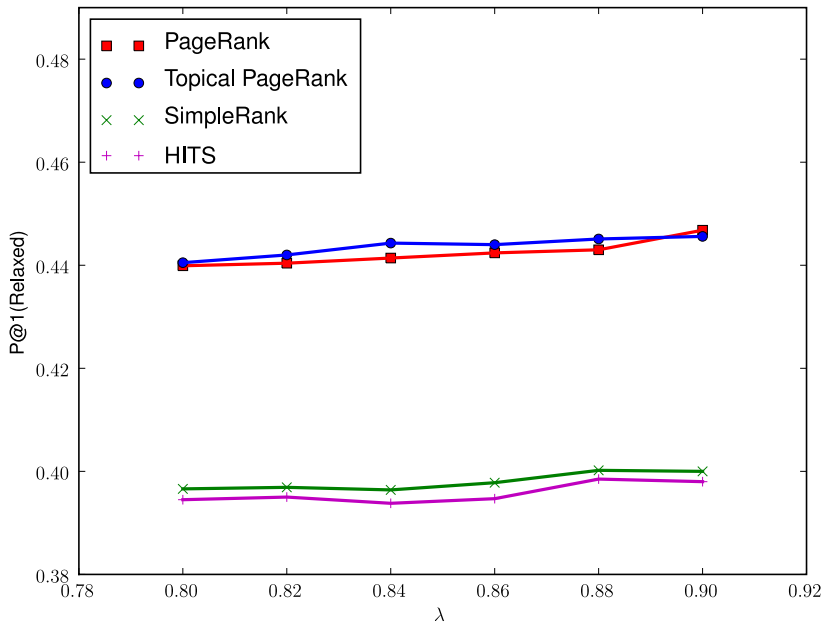

Figure 3.5: P@1(Relaxed)

latent variables (topics) would necessarily cover major topics. However, this number can be given by the number of positive singular values of the word-chunk matrix, a popular technique used in text mining [51].

In all previous figures in experiments, PageRank and Topical PageRank perform similarly and we want to see whether there is a significant difference or not. We perform a t-test on each evaluation metric, showing that Topical PageRank does significantly better than PageRank on P@1(Strict), MRR and P@1(Relaxed) (p-value=0.05) while PageRank does significantly better than Topical PageRank on P@10 and MAP (p-value=0.05). The possible reason that Topical PageRank performs better on those metrics that only consider one result per query is that Topical PageRank can capture the notion that certain users only have expertise on some topics. So a user may not be an overall computer expert

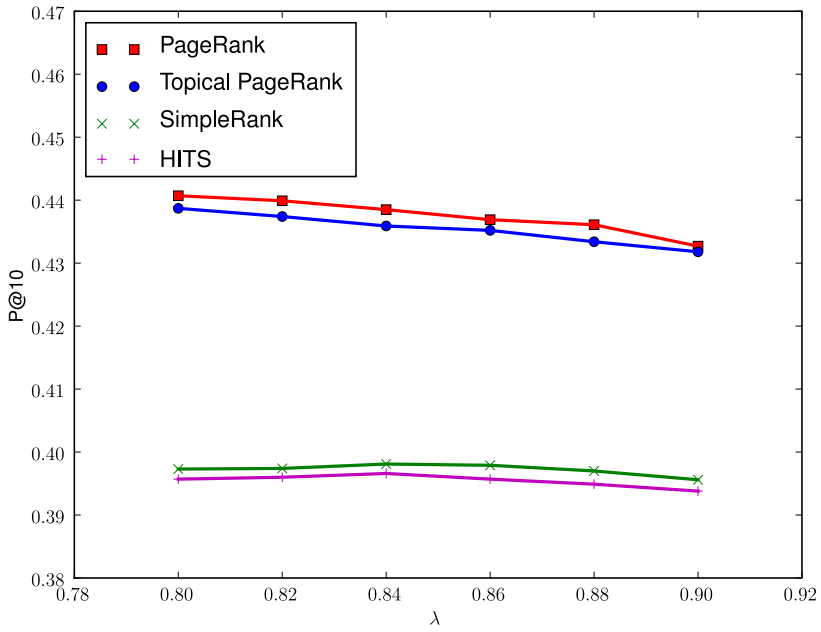

Figure 3.6: P@10(Relaxed)

but can provide several good answers just for hardware repair. In other words, PageRank would average the authority score for all topics and therefore give better approximation for more macro-level evaluation metrics such as P@10 and MAP. Another reason that the results of PageRank and Topical PageRank are close is that our dataset only consists of questions and answers in one main category, Computer & Internet. Compared to [152] where they used topics of the top level of the ODP hierarchy, the difference between topics in our dataset is relatively small. You can imagine that a good expert in Computer may not be an authority in Sports but we probably need to agree that a good expert in Computers may also be an expert in Computer Software. In this case, Topical PageRank shows similar performance as PageRank itself. However, we postulate that if a more topic-diverse dataset is used, Topical PageRank would provide more benefit than PageRank because

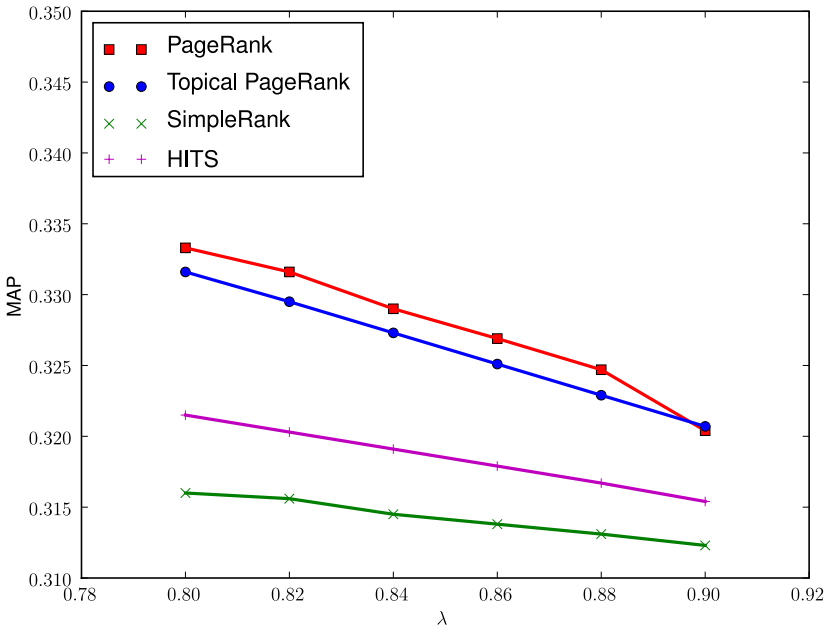

Figure 3.7: MAP(Relaxed)

Table 3.13: Results of different α

α	P@1(S)	MRR	P@1(R)	P@10	MAP
0.70	0.0225	0.0476	0.4450	0.4334	0.3229
0.75	0.0223	0.0475	0.4447	0.4335	0.3229
0.80	0.0223	0.0475	0.4458	0.4336	0.3229
0.85	0.0224	0.0475	0.4460	0.4336	0.3230
0.90	0.0221	0.0473	0.4456	0.4337	0.3230

this less diverse dataset already shows the improvement of Topical PageRank. Since Topical PageRank has a parameter α to indicate the probability of whether to stay on the topic or jump to other topics, we choose several different α values to see how this parameter influences our experimental results. In this set of experiments, we also combine the ranking with BM25 where λ is chosen as 0.88, the value at which the best performance of Topical PageRank is achieved. Table 3.13 details the results. Obviously, different α values

do not influence final ranking results much. We think it is due to the lower diversity of our dataset since "jumping" to other topics would still remain in Computer & Internet.

3.3.4 Summary

In this section, we reviewed two popular ranking schemes, HITS and PageRank, and their applications in user reputation modeling. Due to the naturally bipartite structure of CQA, the HITS scheme and its variations attract more attention in related work and they do improve retrieval performance according to our experiments. On the other hand, we discussed the possibility of using a PageRank-like scheme and introduced topical link analysis into user reputation models. We showed how to incorporate unsupervised topic analysis, in our case pLSA, into topical link analysis. The performance of PageRank and Topical PageRank is much better than HITS and other heuristics in our experiments, which indicates a "hierarchical property" of user reputation. In addition, Topical PageRank is slightly better than PageRank in P@1(Strict) and MRR. We also found that while in general user reputation can help retrieval performance when incorporated with BM25, the performance of returning the exact best answer to a particular question (P@1(Strict) and MRR) decreases. We view this problem as a difficulty that most user reputation models give a "global" reputation for a user which may not reflect the authority of a user in certain topics. Additionally, BM25 plus user reputation models may not be a good approach to the Question Answering task but are good indicators for returning high quality content (which is different from Question Answering!).

3.4 Bibliographic Notes

Although discussion boards are a popular destination for users looking for help, relatively little research directly addresses the problem of mining question answering content from

discussion boards.

Cong et al. [54] was the first to address a problem similar to what we discuss in this chapter. They developed a classification-based method for question detection by using sequential pattern features automatically extracted from both questions and non-questions in forums. They preprocessed each sentence from the first posts by applying a Part-Of-Speech (POS) tagger while keeping keywords including 5W1H (What, When, Why, Where, Which and How) words and modal words. The sequential pattern features are based on the results of the POS tagger. Though achieving reasonable performance, this approach suffers from the typically time-consuming POS analysis process. More importantly, the definition of "questions" in their work is slightly different from our work. They focused on question sentences or question paragraphs while we treat the first post as a whole if it is about a question. For the subtask of finding answers, they proposed an unsupervised graph-based approach for ranking candidate answers leveraging the relevance between replied posts, the similarity between the replied post and the first post, and author information as well. Our method outperforms their approach both in effectiveness and efficiency.

A second related work is that of Ding et al. [60] who proposed a general framework based on Conditional Random Fields (CRFs) to detect the contexts and answers of questions from forum threads. They did not address the question detection subtask in the work and their approach is a complicated method that may not apply to larger datasets. Some features they used within the framework are the same as what we used in this chapter. However, they did not provide a careful comparison of those features and show how different features contribute to the results.

In addition to these two directly related papers, there is some research on knowledge acquisition from discussion boards. Zhou and Hovy [228] presented a summarization system utilizing the input-reply pairs extracted from online chat archives. Their system is not specifically designed for question answering content. Feng et al. [71] proposed

a system to automatically answer students' queries by matching the reply posts from an annotated corpus of archived threaded discussions with students' queries, which is a different problem from our work. Huang et al. [101] presented an approach for extracting high-quality ¡thread-title, reply¿ pairs as chat knowledge from online discussion boards so as to efficiently support the construction of a chat-bot for a certain domain. They also did not focus on question related threads in discussion boards.

Other previous work tried to understand and mine discussion boards for more general purposes. Antonelli and Sapino [12] proposed a system to classify discussion threads based on rules derived by using both speech acts and graph analysis. Although their system can identify questions and answers as well as other types of threads, their dataset was small and they only provided precision measures in their experimental results. Kim et al. [114, 113] and Feng et al. [72] used speech acts analysis to mine and assess discussion boards for understanding students' activities and conversations. They used only a small dataset and did not address question answering content in their work. Lin and Cho [134] introduced several techniques to preprocess questions extracted from discussion board including "garbage text" removal, question segmentation and merging questions. They did not discuss how to identify question content and their answers. Shrestha et al. [179] detected interrogative questions using a classification method and built a classifier to find answers using lexical features based on similarity measurement and email-specific features.

Compared to the problem we address, extensive research has been done on QA services like Yahoo! Answers or other Frequent Asked Questions (FAQ) services. Jeon et al. [106, 105], Duan et al. [62], and Cao et al. [35] tackled the problem of finding questions in the QA services that are semantically similar to a user's question. Song et al. [185] proposed a metric "question utility" for studying usefulness of questions and showed how question utility can be integrated into question search as static ranking. Jeon et al. [107] presented a framework for using non-textual features like click counts to predict the quality of answers,

incorporated with a language modeling-based retrieval model. Surdeanu et al. [187], Xue et al. [212], Berger et al. [22], Jijkoun et al. [108], and Riezler et al. [169] described various retrieval models or systems to extract answers from QA or FAQ services. Liu et al. [138] proposed automatic summarization techniques to summarize answers for re-use purposes. Gyongyi et al. [80] performed an analysis of 10 months of Yahoo! Answers data that provided insights into user behavior and impact as well as into various aspects of the service and its possible evolution. Some of the above work is complementary to our approach, and therefore could be employed to enhance our methods but in general all work above does not need to detect questions.

Traditional Question Answering tasks in TREC style have been well studied; see for example Vorhees [192]. That work mainly focused on constructing short answers for a relatively limited types of questions, such as factoid questions, from a large corpus. This makes it possible to identify the answer type. In contrast, typical questions extracted in discussion boards are more complex and usually consist of multiple sentences or even several paragraphs, and it is also difficult to represent and identify answer types for those questions.

Community-based Question Answering (CQA) has become an active research area. Much of the work has focused on Yahoo! Answers due to its popularity. Song et al. [185] propose a metric "question utility" for studying usefulness of questions and showed how question utility can be integrated into question search as static ranking. Various retrieval models or systems had been proposed (e.g., [187, 212]) to extract answers from QA or FAQ services. Jeon et al. [107] present a framework for using non-textual features like click counts to predict the quality of answers, incorporated with a language modeling-based retrieval model. Agichtein et al. [4] presented a supervised approach to mining user interaction and content-based lexical features to identify high quality content in CQA. Bian et al. [23] develop a ranking system to retrieve relevant and high-quality answers. Most

models above do not explicitly integrate content quality and user reputation information into the ranking process. Hong and Davison [93] show that user authority is a good indicator for retrieving answers from discussion boards. Zhang et al. [223] applied both ExpertiseRank and HITS to identify users with high expertise. Jurczyk and Agichtein [110] show an application of the HITS algorithm to a CQA portal, especially the user interactions graph, and show a positive correlation between authorities calculated with the HITS algorithm and answer quality. Zhou et al. [227] propose a method for co-ranking authors and their publications using their networks. Most of the works discussed above do not provide a comparative study of how their ranking scheme outperforms others. At the same time, most ranking schemes are based on the HITS algorithm.

Two of the most prominent link analysis algorithms, PageRank [153] and HITS [118], have been shown to be successful in the context of evaluating quality of Web pages. Nie et al. [152] proposed Topical PageRank and Topical HITS which embed topical information when propagating authority scores. They showed that topical PageRank and topical HITS outperform PageRank and HITS respectively. As far as we know, no research work has shown whether these ranking schemes can be applied to user reputation modeling especially in the context of CQA.

Chapter 4

Global Information Filtering in Twitter

4.1 Introduction

In the previous chapter, we explored basic information filtering in CQA portals. Although CQA portals are vital to meet numerous users' information need, micro-blogging sites have emerged as an important alternative type of online conversational media for users to share information in recent years. Starting from this chapter and following several chapters, we focus on the problem of information filtering in micro-blogging services, and Twitter in particular. Social network services such as Facebook, Myspace and Twitterhave become important communication tools for many online users. Such websites are increasingly used for communicating breaking news, eyewitness accounts and organizing large groups of people. Users of these websites have become accustomed to receiving timely updates on important events, both of personal and global importance. For example, Twitter was used to propagate information in real-time in many crisis situations such as the aftermath of the Iran election (June, 2009), the tsunami in Samoa (September, 2009) and the

Haiti earthquakes (January, 2010). Many organizations and celebrities use their Twitter accounts to connect to customers and fans.

The social aspect of Twitter and similar websites is given by the fact that each user can connect to other members, forming a network of relationships. In Twitter, the network of connections is a directed graph and thus users can receive updates from accounts which would not otherwise reciprocate the relationship. An important characteristic of micro-blogging websites is that users receive messages strictly from their direct friends (whom they are following). Therefore, the amount of information as well as the overall quality of these messages one user receives would highly depend on whom he or she follows. Some problems may arise due to this paradigm of information flows induced by the social graph. For those users who follow a large number of friends, the amount of messages they receive daily would easily exceed their capability to handle, causing the problem of information overload. Hence, in this case, it is crucial to filter out trivial messages while leaving the ones that users really care about. On the other hand, for those users who do not have enough friends to follow, important messages will hardly reach them, potentially making users gradually lose interest in the service. Therefore, it is useful to recommend some interesting and popular messages to users, possibly attracting them to follow new users. Both problems require us to determine the value of individual messages and their popularity.

In Twitter, popular messages are mostly shared among users in two forms. One is that users can propagate important or interesting messages in the social graph by "retweeting" other people's messages. A retweet is normally a copy or a variant of the original message, sometimes with the username of the original author appended to it. This type of message is vital for the flow of information within the network, as users can get exposed to ideas recommended by their friends. Messages can "travel" long distances in the social graph through a series of retweets, forming a retweet tree that starts from the original author

of the message, described in Kwak et al. [125]. It is common for popular messages to be retweeted thousands of times in a very short period of time. Another form of popular messages is that users share external URLs by posting links messages because of the length limitation of Twitter messages. In this case, if a link is posted by different people numerous times, we would consider that the link and the information behind the link become the keystone for the propagation. We denote this form as "URL sharing". Thus, the problem of determining the value of a message can be converted into the question of whether a message will be retweeted, and whether the URL in a message will be shared in the future. We address both problems by utilizing machine learning techniques to predict if and how often new messages will be retweeted and a URL will be shared in the future. We believe that a system that can predict such messages could be able to determine messages with potential to be become breaking news. In addition, those popular messages could be delivered earlier to the users, thus short circuiting and speeding the delivery of relevant information. Studying the propagation of popular messages can provide insights on both the social network and on particular individuals. In addition, we can view popular messages and which reach many parts of the network to be important to the community as a whole. Also, at an individual level the popular messages of users can shed light into their interests. It is therefore valuable to determine the characteristics of messages which become popular as opposed to messages which get lost in the sea of information.

Our approach to tackle the problem is to treat it as a classification task. We train classifiers with positive and negative examples of messages which will be retweeted in the future and which contain URLs which are shared in the future. To build such kind of classifiers, we investigate a wide spectrum of features to determine which ones can be successfully used as predictors of popularity, including the content and topical information of messages, graph structural properties of users, temporal dynamics of popular messages and meta-information of users and messages as well. Our experiments are conducted on

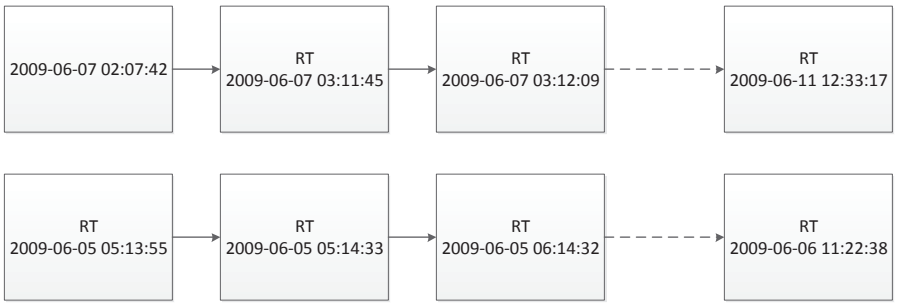

Figure 4.1: "Retweet Chains"

two massive real-world datasets and the results suggest that we can successfully predict whether a message will be popular or not and its volume with good predictive performance. In this chapter, we try to answer the following questions:

- What features are useful for predicting popular messages?
- Are the features for low volume popular messages the same as the ones with high volume?
- Are the popular messages predicted from our method "legitimate messages", or just spam?

The chapter is organized as follows. In Section 4.2, we introduce the notion of popular messages in Twitter. In Section 4.3, we explore a wide range of features that can be utilized to tackle the questions mentioned above. In Section 4.4, we demonstrate the effectiveness of our proposed method in multiple datasets. Related work is discussed in 4.6 and we conclude this chapter in Section 4.5.

Table 4.1: Sample "retweet chain".

how to: build your personal brand on youtube http://bit.ly/2uhunr
rt @mashable how to: build your personal brand on youtube http://bit.ly/3arqzh
rt @jimgaffigan: rt @mashable: how to: build your personal brand on youtube http://bit.ly/3arqzh

4.2 Popular Messages in Twitter

We consider the problems of predicting two types of popular messages, "retweets" and "URL sharing". Both problems require an effective way to build datasets to be investigated. Since our datasets are only a sample of actual Twitter messages, we cannot recover all examples of two types of information flows. In other words, for "retweets", we do not always know which message is exactly a retweet of another message. For "URL sharing", we do not know how many URLs in the dataset appear in the complete set and therefore cannot predict the popularity accurately. However, we argue that even if someone may have the complete set of messages, it is still a question to recover exact patterns, especially for "retweets". For example, it is difficult to determine the origin of all retweets perfectly. Due to the character limit of Twitter messages, many users may need to modify the original message in a variety of ways to retweet a message. Therefore, a retweet may not exactly match the original message. Moreover, some users may add some personal opinions and feelings in retweets, which make the process of identifying the original message more difficult. Here, we describe a simple method to construct a meaningful dataset for "retweets" as follows, while a similar process is also applied to "URL sharing". Note that because we cannot obtain retweets by the newly developed Twitter API[1] for the datasets we experiment on. Conceptually, we want to build "retweet chains" as Figure 4.1.

First, we pre-process the datasets by removing links from the messages, removing any word starting with the "@" character, removing messages containing non-latin characters

[1]http://dev.twitter.com/doc/get/statuses/retweets/:id

and converting all characters to lower case. The term "RT" is temporarily removed from messages and MD5 scores are calculated for the remaining strings. Two messages are defined as "identical" if they share the same MD5 value. We sort all "identical" messages in ascending time order, forming many chains of messages. The first message in the chain is considered to be the original message, even if in reality this might not be the case, as described above. We require that all later messages in the same chain should contain at least one "RT" term, indicating that they are retweets, and discard all messages without "RT" terms. Therefore, it does not matter if the first message in the chain is a retweet, but all subsequent messages in the chain must be retweets. Table 4.1 demonstrates one example of of "retweet chains" in our dataset where each line is a message while the top one is considered to be the original message. For "URL sharing", we do the same preprocessing steps but leave the links in messages. We build URL chains (one for each link) by grouping all the messages contain the same URL and sort them by ascending time order. Note, we do not resolve all shortened URLs and we do not care that whether a message contain "RT" or not in this case.

Using the "retweet chains" and "URL chains" built as above, four general questions (with their abbreviation in the parentheses) are tackled in this chapter under a classification framework regarding each message:

- Whether or not the message will be retweeted. (`Q1`)
- Whether or not the message will be retweeted with a certain volume. (`Q2`)
- Whether or not the URL in the current message will be shared in the future. (`Q3`)
- Whether or not the URL in the current message will be shared in the future with a certain volume. (`Q4`)

By studying these problems, we wish to unveil some specific characteristics of popular messages and how these messages can be differentiated from trivial messages. For `Q1`, if

there are n messages in a particular chain, we take the first $n-1$ messages as "positive instances", which means that they will be retweeted in the future at least once, and the last one as a "negative instance". In addition, all other messages which are not in any chains are also considered as "negative instances". For `Q2`, we set a certain threshold τ and treat messages with the number of future retweets above τ as "positive instances" and all other messages as "negative". We will investigate the change in performance and effectiveness of features as the threshold is varied. For `Q3`, the messages which contain URLs that are shared in the future are considered to be "positive instances" while all other messages as "negative". `Q4` is treated similarly as `Q2` where we have a sliding threshold for classification.

Note, we recognize some false positives in "retweet chains". For example, although many messages are exactly the same (e.g., "happy birthday") and contain "RT" as well, through some manual validation, we found that they are definitely not retweets with the same origin. Since "retweet chains" are constructed artificially, we further validate all messages by the connections between their authors. More specifically, we verify whether there exists a connection between the authors of two consecutive messages in the chain. If there is a follower-followee connection, we consider the later message to be a "real" retweet of the previous one. Otherwise, we remove the later message. By filtering messages in the chain in this way, we also construct smaller validation datasets, denoted as "constrained datasets" and perform classification tasks on these datasets.

4.3 Features

In this section, we discuss a set of features that are applicable for questions `Q1`-`Q4` in detail. We group the features into content features, structural features, temporal features and meta-data features, as shown in Table 4.2. There are at least two reasons for using

Table 4.2: Features

Type	**Name**	**Abbreviation**
Content	TF-IDF of terms in message	*TFIDF*
Structural	# of followers	*Indegree*
Structural	# of friends	*Outdegree*
Structural	# of followers of followers	*SecondLevel*
Meta	whether the message has been retweeted	*MsgRT*
Meta	# of the message have been retweeted	*MsgRTNum*
Meta	# of messages are retweeted by author	*UserRT*
Meta	whether the URL has been shared	*URLShare*
Meta	# of the URL have been shared	*URLShareNum*
Meta	# of URLs in the message	*URLNum*
Meta	# of hashtags in the message	*HashtagNum*
Meta	# of mentions (@) in the message	*MentionNum*
Meta	Tweets contain URLs per user	*URLTweetsUser*
Meta	hashtags per words	*HashtagsWords*
Meta	URLs per words	*URLsWords*
Temporal	the time elapse since the origin	*SinceOrigin*
Temporal	the time elapse since the previous	*SincePrevious*
Temporal	average time elapse of messages	*AverageBefore*
Temporal	average time elapse of a user's messages	*AverageUser*

the content of messages to predict their popularity. First, certain popular topics may influence people to share and propagate information by retweeting. Second, by using the content information we wish to model users' interests and see whether they influence the propagation of information. Here, we use Bag-of-Word (BOW) representation for the terms in messages and calculate their TF-IDF scores. Alternative representations which are more complicated may be considered, such as the sparse bi-gram model used in Lee et al. [128] to detect Twitter spam and the topic model representation used in Hong et al. [92]. Here, we use TF-IDF for simplicity.

Besides the content of messages, we also believe that the social relationships of a user play a crucial role in the process of information propagation. Here, we denote the features related to users' social graphs as structural features. Three structural features are used in

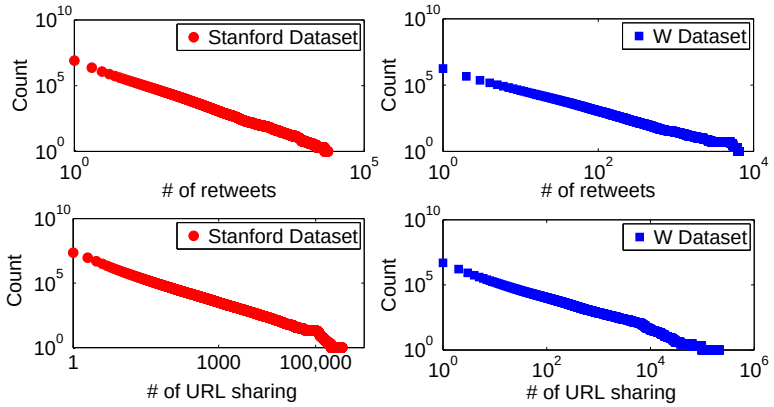

Figure 4.2: The distribution of retweets and URL sharing.

this paper: the number of followers and the number of friends one user has, and the number of followers of followers one user has (essentially, the number of second level followers). Due to the system design of Twitter, users mainly see the messages from their direct friends. Therefore, messages written by a user with zero in-degree are highly unlikely to be retweeted in the future. Moreover, a user with out-degree zero (no friends) has limited information sources and thus may not be able to pass other popular or important messages to her followers. So, we hypothesize that these structural features might play an important role in determining popular messages. Some other features, like PageRank and local clustering coefficient [19], which were not effective in our preliminary experiments, are excluded in this work. Most Twitter messages are time sensitive, meaning that they are valuable only during a short time period. For a particular message, users may either eventually receive the message from different retweet paths if available or do not receive the message due to its triviality. In other words, messages might be out of date very quickly, especially the ones related to current events and news. Therefore, retweets may have a strong relationship with temporal information. Several temporal features are considered and all of these features are based on the chains constructed as described in the previous section. Suppose we have K retweet chains in total. For each chain, we use $t_{k,i}$ to denote

the time stamp of message i in the chain k where $i \in \{0, ..., n\}$ and $t_{k,c}$ to denote the time stamp of the current message. We also use $t_{k,c}^{(u)}$ to denote the author of the current message is user u and N_u as the number of times user u's messages get retweeted.

- **Since Origin**: The time difference between the current message and the origin: $t_{k,c} - t_{k,0}$
- **Since Previous**: The time difference of current message and the previous one: $t_{k,c} - t_{k,c-1}$
- **Average Before**: The average time difference before current message in the same chain:$\frac{1}{c}\sum_{i=1}^{c}(t_{k,i} - t_{k,i-1})$
- **Average Time Per User**: For a particular user u, $\frac{1}{N_u}\sum_{k=0}^{K}\sum_{i=0}^{c}(t_{k,i+1}^{(*)} - t_{k,i}^{(u)})$

All these features may capture different types of aspects for retweets. More specifically, the first two features measure the freshness of the message. If a message is relatively old, other users may have already seen it and therefore the probability of it being retweeted is likely to diminish. The third feature measures the basic propagation speed of the message. Different types of messages have different propagation speeds. For instance, breaking news may have faster speed of propagation than some messages related to a week-long conference (e.g., WWW or SIGIR). The last feature regarding temporal information is related to the author of the target message. If an author produces messages that usually get retweets in a short time period, this would be an indicator that the messages generated by this user will also be retweeted in the future. Note, these features can be also calculated for "URL sharing" chains.

Meta information related to messages and authors can be very helpful. First, we consider whether the message has been retweeted before. Although we are predicting the possibility that this message will be retweeted in the future, the history of this message

Table 4.3: Statistics about the datasets

W Dataset	
Total messages:	269,466,135
Total "retweet" chains:	1,782,948
Total messages in all chains:	5,482,914
Stanford Dataset	
Total messages:	472,126,381
Total "retweet" chains:	8,083,011
Total messages in all chains:	26,529,634
W Dataset (Constrained)	
Total "retweet" chains:	262,612
Total messages in all chains:	1,174,127
Stanford Dataset (Constrained)	
Total "retweet" chains:	1,778,262
Total messages in all chains:	10,081,260
KAIST Graph	
Number of user's connections:	1.47 billion
Number of distinct users:	41.7 million

may be a strong indicator of its future popularity. We also consider whether the messages a user produces have ever been retweeted before as a feature. If a user usually generates messages that are retweeted, the new messages by the same user might has higher chance to be retweeted. Other meta features are straightforward that have been already used in similar papers, such as the number of URLs in the message and the number of hashtags in the message and other variants.

4.4 Experiments

Two datasets are used in our experiments. We collected the first using Twitter's Streaming APIs[2], denoted as "W Dataset", consisting of messages from November and December 2009. Another dataset, denoted as "Stanford Dataset", is obtained from Yang and

[2]http://dev.twitter.com/

Table 4.4: Performance on question Q1

W Dataset			
Feature set	**Precision**	**Recall**	**F_1**
TFIDF	0.744	0.629	0.682
Non-content	0.896	0.850	0.872
TFIDF + Non-content	0.899	0.863	0.881
Stanford Dataset			
Feature set	**Precision**	**Recall**	**F_1**
TFIDF	0.722	0.650	0.683
Non-content	0.859	0.879	0.869
TFIDF + Non-content	0.860	0.895	0.877
W Dataset (Constrained)			
Feature set	**Precision**	**Recall**	**F_1**
TFIDF	0.522	0.097	0.164
Non-content	0.530	0.101	0.170
TFIDF + Non-content	0.643	0.157	0.252
Stanford Dataset (Constrained)			
Feature set	**Precision**	**Recall**	**F_1**
TFIDF	0.542	0.113	0.187
Non-content	0.562	0.121	0.200
TFIDF + Non-content	0.668	0.189	0.295

Leskovec [213], consisting of messages from June, 2009 to December, 2009. For both datasets, we remove all messages containing non-ASCII characters from the messages and only keep the messages with at least five words, in addition to links and hashtags. For connections between users, we use the graph from Kwak et al. [125]. More detailed statistics are shown in Table 4.3. We plot the distributions of "retweets" and "URL sharing" in log-log scale for both datasets in Figure 4.2. The X-axis is the tweets or URLs with corresponding number of sharing or retweets. The Y-axis is the counts of such kind of tweets. Note, all sub-figures suggest heavy-tailed distributions for "retweets" and "URL sharing". Taking "retweets" for instance, most tweets that are retweeted only attract one retweet while a small number of tweets can receive a large number of retweets.

In our experiments, we need a classifier that can be trained and tested on millions

of instances with millions of features. Plus, since the number of positive instances is significantly smaller than the negative instances, a classifier that optimizes for accuracy or error rate may not be appropriate for our task. We choose Logistic Regression as our classifier, which is a linear classifier and can use arbitrary numerical values as features. The implementation used is Logistic Regression with l_2 regularization[3]. In order to overcome the problem of unbalanced data, we sub-sample the negative instances down to the similar size of positive ones for training but retain the complete data set for testing.

Since the Twitter data has strong temporal effects, classifiers may gain additional advantages if cross-validation style evaluation is performed. Indeed, in preliminary cross-validation experiments, only a *TFIDF*-based classifier may achieve very high F_1 scores. Thus, to be more realistic, we mimic the settings that might be used in real applications, adopting a "semi-online" evaluation method. We train the classifier on one week or one month data and test it on the next week or month. We do it iteratively for all the weeks or months in our datasets. The features in a particular week or month are generated only based on previous weeks or months. In this case, we do not give the classifier additional hints of the "future". More specifically, for "W Dataset", we train and test the classifier on weekly basis. For "Stanford Dataset", we train and test the classifier on monthly basis. Results are averaged across those weeks or months. Standard metrics including Precision, Recall and F-measure are used in the experiments.

4.4.1 Predicting Popular Messages

For question `Q1`, whether or not a message will be retweeted in the future, the results are shown in the upper part of Table 4.4. We conduct experiments on three different settings: 1) only based on *TFIDF*; 2) only based on all non-content features (*Structural* + *Meta* + *Temporal*); and, 3) combine *TFIDF* and all non-content features. The performances are

[3]http://www.csie.ntu.edu.tw/~cjlin/liblinear/

Table 4.5: Top ranked features by χ^2 scores for `Q1`

W Dataset		**Stanford Dataset**	
Rank	Feature	Rank	Feature
1	*MsgRTNum*	1	*MsgRTNum*
2	*AverageBefore*	2	*MsgRT*
3	*SincePrevious*	3	*SinceOrigin*
4	*SinceOrigin*	4	*AverageBefore*
5	*MsgRT*	5	*SincePrevious*

comparable on both datasets even though "W Dataset" is trained and tested on weekly basis while the "Stanford Dataset" is trained and tested on monthly basis, suggesting that the features are quite stable across the datasets and timeframes used for evaluation.. Although the best performance is achieved by combining all the features, it is clear that a classifier only based on non-content features may be sufficient.

As mentioned in Section 4.2, the "retweet chains" we constructed are artificial chains. Therefore, we further experiment on the "Constrained datasets" in which the users of two consecutive messages in the chain are connected on the Twitter graph. The results are shown in the bottom part of Table 4.4. Note, the overall performance drops dramatically as expected since both datasets are a smaller, sparser sample of all tweets in Twitter and the graph is also incomplete. We see that in this case, the combination of content features and non-content features still performs well compared to only non-content features. Overall, the results on original chains may effectively give an "upper bound" of the performance on question `Q1` while the results on strict chains give us "lower bounds".

In addition to the classification performance, we calculate the correlation between features and class labels by Chi-square tests (χ^2 test) provided in WEKA [81] on all non-content features since they are more interesting than single terms in the task. The top five ranked features from both datasets are shown in Table 4.5. It is clear that there is a consensus between two datasets on the top five features although the order of

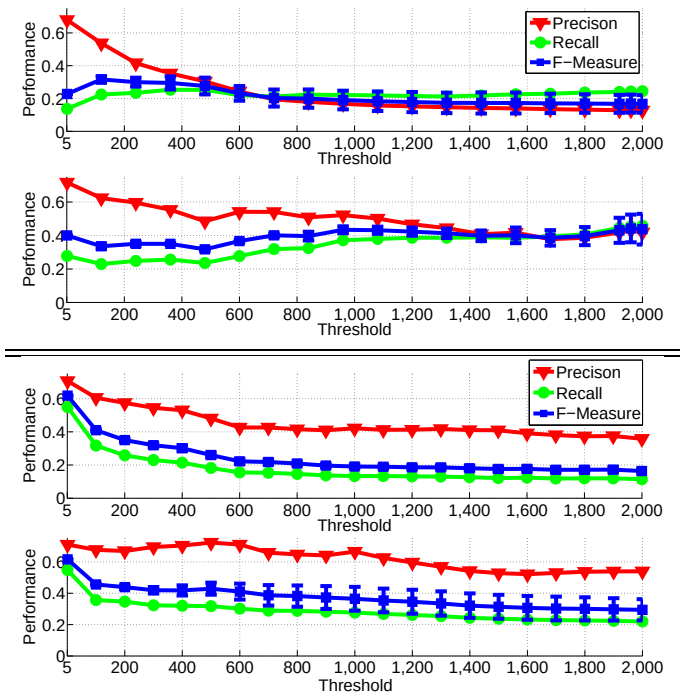

Figure 4.3: The performance of sliding threshold for question `Q2`.

ranking is not identical. Basically, whether a message has already been retweeted before and several temporal features are strong indicators for the messages to be retweeted in the future. For temporal features, it is not unexpected that they highly correlate with positive instances since these features are empty for most negative instances which are not in any "retweet chains". For "retweet chains", we conduct two-sample Kolmogorov-Smirnov tests (KS test), a popular technique to detect whether feature values for different instances are from the same underlying distribution (e.g., Becchetti et al. [19]). For these three features, we gather the empirical distribution of feature values on positive instances and negative instances, conducting the KS test on the two distributions. All the tests reject the null hypothesis that the feature values for positive instances and those for negative instances are from the same underlying distribution. For *AverageBefore*, the Kolmogorov-Smirnov difference (KS difference) is 0.0265 and p-value for rejecting the null hypothesis

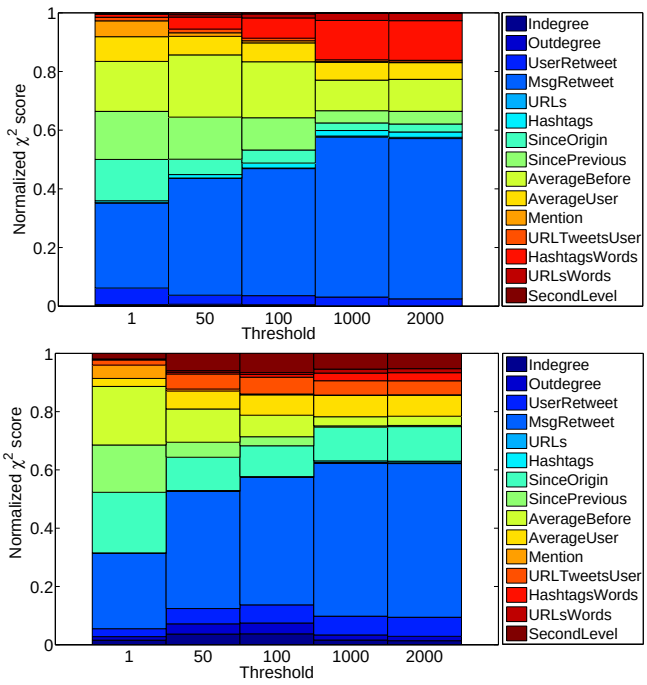

Figure 4.4: Feature ranking changes of different thresholds.

equals to $1.4705 \cdot 10^{-282}$. For *SincePrevious*, KS difference is 0.0544 while p-value equals to $3.2203 \cdot 10^{-276}$. For *SinceOrigin*, KS difference value equals to 0.0816 and p-value is $6.0070 \cdot 10^{-28}$. In other words, a classifier built on these features can take advantage of the fact that the empirical distribution of feature values among positive instances and negative instances are from different underlying distributions.

In real-world applications, whether a message will be retweeted in the future may not be so interesting since users are interested in popular messages, the ones repeated numerous times by others. For this question, we use a slide threshold method to conduct experiments on a series of binary classification tasks. We set the threshold τ, representing the number of retweets a message will receive in the future. All the messages receiving more than τ retweets are positive instances and all others are negative instances. We move this threshold from 1 to $2,000$, which is a fairly large number in our datasets. The results

on "W Dataset" (upper) and "Stanford Dataset" (lower) are shown in Figure 4.3 while both figures contain two sub-figures. The standard deviation of F_1 values are shown as error bars in the figures. Three basic conclusions can be made through the results. First, the performance (the overall F_1 values) is worse than it was in `Q1` in which the threshold was effectively 1. As shown in Figure 4.2, there are large number of messages that only receive low volume of retweets. Thus, a threshold larger than 5 (inclusive) may further cause the datasets to be even more skewed, making the prediction problems harder for the classifier. The second observation is that the combination of non-content features and *TFIDF* perform more robustly than only using non-content features while this is not obvious in `Q1`. Additionally, the standard deviation of F_1 scores increases as the threshold increases, indicating that performance varies from week to week (or month to month) dramatically. The effectiveness of features is not stable over time. Since, in this work, only simple temporal features are utilized, we hypothesize that features that consider the trends of content and topics may help with this situation.

For feature ranking, we take the scores obtained by χ^2 test and normalize them into $0 \sim 1$. Therefore, for all threshold values, the correlation between features and class labels are compatible. We plot this proportional correlation for "W Dataset" (upper) and "Stanford Dataset" (bottom) in Figure 4.4 over five different threshold choices. The upper sub-figure shows the performance of non-content features while the bottom sub-figure shows the performance of the combination of *TFIDF* and non-content features. The first conclusion is that the proportional correlation between features and class labels is drastically different for low volume retweets versus high volume retweets. For low volume retweets, several features (e.g., *MsgRT*, *AverageBefore*, *SinceOrigin*, *SincePrevious*) are approximately equally correlated to the class label and they are consistent on two datasets while *MsgRT* dominates in later threshold values and some features tend to be more correlated with class labels on either one dataset (e.g., *AverageBefore* for "W dataset"

Table 4.6: Performance on question `Q3`

W Dataset			
Feature set	**Precision**	**Recall**	**F_1**
TFIDF	0.808	0.747	0.776
Non-content	0.837	0.893	0.864
TFIDF + Non-content	0.886	0.868	0.877
Stanford Dataset			
Feature set	**Precision**	**Recall**	**F_1**
TFIDF	0.788	0.753	0.770
Non-content	0.842	0.878	0.860
TFIDF + Non-content	0.888	0.880	0.884

Table 4.7: Top ranked features for question `Q3`

W Dataset		**Stanford Dataset**	
Rank	Feature	Rank	Feature
1	*URLShare*	1	*URLShare*
2	*URLAverageBefore*	2	*URLSinceOrigin*
3	*URLSinceOrigin*	3	*URLAverageBefore*
4	*URLSincePrevious*	4	*URLSincePrevious*
5	*URLsWords*	5	*URLsWords*

and *SinceOrigin* for "Stanford dataset"). This may partially explain that the performance for `Q1` on two datasets are very similar but it is noticeably different on `Q2`, shown above in Figure 4.3. Overall, it seems that temporal features are correlated to all threshold values although the contribution of them may vary. Note, since we normalize the χ^2 scores just for visualization purposes, the proportion for different thresholds are not really comparable. A small proportion does not indicate that this feature has little contribution in the classification performance. Similar to `Q1`, we also investigate the problem of whether a URL in the current message will be shared in future or not, regardless of its volume. The results are shown in Table 4.6. Again, the combination of non-content features and *TFIDF* is better than using non-content features only, though the margin is small, as in `Q1`. Using the same feature ranking techniques, we see that temporal features are significantly

correlated to the class label. A URL that has been shared before (*URLShare*) will strongly indicate that this URL may be shared in the future. The feature rankings are also similar on the two datasets.

We also slide the threshold of the number of sharings a URL may receive in the future to investigate how this classification-based method performs for larger volume URL sharing. The results on "W Dataset" (upper) and "Stanford Dataset" (bottom) are shown in Figure 4.5. Both figures contain two sub figures. The upper sub figure shows the performance of non-content features while the bottom sub figure shows the performance of the combination of *TFIDF* and non-content features. The standard deviation of F-measure are shown as error bars in the figures. As in `Q2`, the performance drops for larger volume but two datasets have more similar performance in this task than it was in `Q2`. Overall, the combination of features achieve more robust and better performance on both datasets.

Similar to `Q2`, we also conduct a feature ranking process to understand the changes of correlations between features and labels. The results on "W Dataset" (upper) and "Stanford Dataset" (lower) are shown in Figure 4.6. Unlike `Q2`, here the results are more stable on both datasets. The temporal features for URLs play significant role as suggested. Additionally, the meta feature *URLShare*, which essentially indicates that a URL has been already shared, strongly correlated to class labels, especially for high volume of sharing URLs.

4.4.2 Spam or Not

Although it would be nice to know that the popularity of messages can be predicted, it is certainly better to know whether the information conveyed by these popular messages is high quality or not. Here, rather than directly assessing the quality of messages, we look at whether the author of a particular message is questionable, namely a spammer.

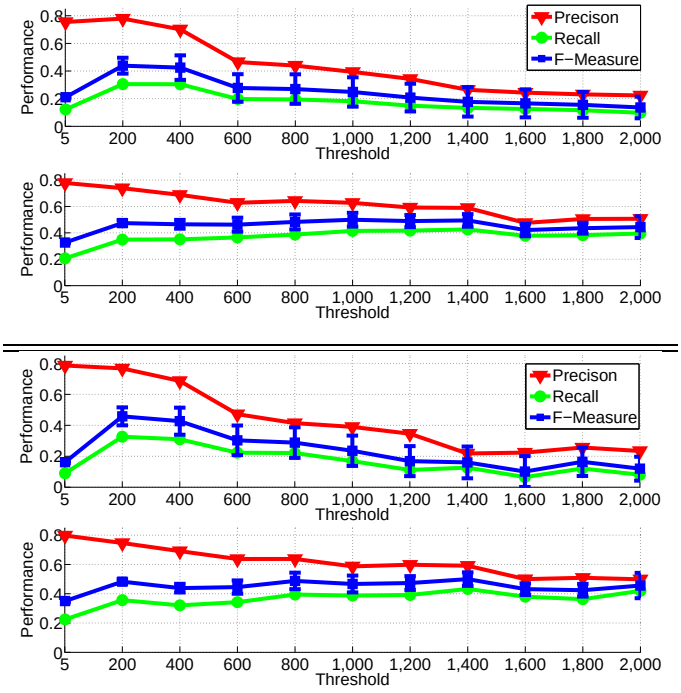

Figure 4.5: The performance of sliding threshold for question `Q4`.

However, it is also a hard question to determine whether a user is a "spam user" due to both the problem of definition and other practical issues (e.g., how to differentiate "spam users" from "legitimate business promoters"). We use a simple method to determine the legitimacy of a user by verifying the existence of the account one year later (in December 2010). Since some spam user accounts might be terminated by Twitter, according to its rules[4], we believe that it is at least reasonable to assume the content generated by these users is suspicious.

Taking `Q1` as an example, for the "W dataset", we found that within a total of 12,380,678 users, 1,040,672 user's accounts (8%) disappeared by December 2010. These suspicious (possibly spammer) users generated 26,854,749 messages, out of which 431,224 are labelled as positive in the dataset (approximately 7.8% of all positive instances). In

[4]http://support.twitter.com/entries/18311-the-twitter-rules

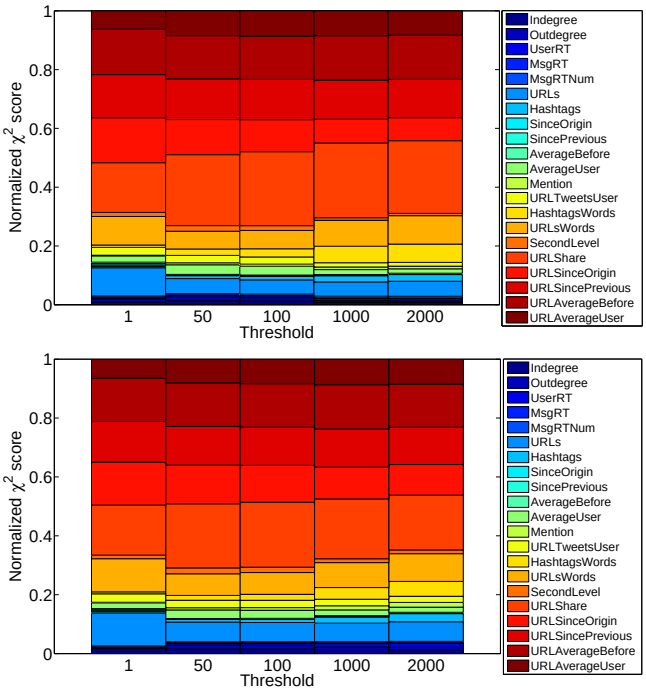

Figure 4.6: Feature ranking changes of different thresholds for question `Q4`.

order to determine the impact of messages from suspicious users, we conduct two simple experiments. If we train on the complete dataset (including these questionable messages) and test on non-suspicious messages, the overall F1 score is 0.8803. In addition, if we only train the classifier on the remaining (non-suspicious) data but test on all messages, the overall F1 score is 0.8801. Both scores are comparable to the one reported in Table 5, suggesting that messages from possibly suspicious users were not affecting classification performance. Similar performance is also achieved on the "Stanford Dataset". However, we do believe that a thorough study of such users and the messages generated by them, as well as a more detailed performance analysis, would be valuable future work.

4.5 Summary

In this chapter, we studied the problem of identifying popular messages from Twitter by employing a classification approach with extensive experiments. Our experimental results demonstrated that non-content features can be effectively used to predict popular messages in general while integrated with content features would give more robust performance on the problem of predicting high volume popular messages. We also conducted a feature selection process on different levels of popularity of messages, showing that temporal features are vital for both "retweets" and "URL sharing" problems.

In addition, we opened a discussion on how spam users potentially influence the propagation of popular messages by investigating the "suspicious messages" produced by questionable users. The preliminary results showed that the current classifier can filter out reasonable amount of these messages while a comprehensive study is required for the future work. The features demonstrated in this chapter can be computationally inexpensive in two senses. First, most features only require a single pass of current data stream and statistics of feature values can be updated on the fly (e.g., *MsgRT*). Secondly, all features can be computed in parallel, and indeed are generated within Hadoop, an open source MapReduce framework [58], in our experiments. Our "semi-online" evaluation settings also suggest that the techniques introduced in this chapter would also be applicable to real-world systems.

4.6 Bibliographic Notes

Twitter has attracted a lot of research attention in the past a few years. Here, we only review the work that attempts to understand the factors influencing popular messages and how messages are propagated on Twitter. One recent study conducted by Suh et al. [186] demonstrates what kinds of factors contribute to number of retweets for a message. The

authors conducted Principle Component Analysis (PCA) on a dataset consisting of 10K messages with nine features and demonstrated how different features were related to latent factors found by PCA and further studied the features on a larger dataset. Although it is interesting to know how effective one feature might be, the authors do not perform any prediction task on the dataset. In their study, they claimed that URLs and hashtags highly correlate with the potential of messages to be retweeted. Similar work is performed by Ye and Wu [217] who use some simple structural features to understand how messages propagate in Twitter network. In a recent poster by Hong et al. [92], we presented some of this material. In addition to retweet counts, however, we refine those ideas, adds a second type of popular message, investigate the effect of a variable threshold rather than an arbitrary set of retweet prediction counts, and comprehensively investigate a somewhat different set of features.

In addition to these experimental approaches to the problem, Yang and Leskovec [213] proposed an adaptive wavelet-based clustering method to characterize temporal variations of social media and conducted their method on Twitter data. Although their work offers some insights of information propagation, they do not address the prediction task. In general, the set of features and the datasets considered in this chapter are much larger than existing work.

One work relevant to our general research problem of building a personalized message recommendation system is that of Chen et al. [46], which describes a combination of methods for recommending popular URLs from Twitter messages. They select a set of candidate URLs from the local graph of each user, then rank them by topic relevance and social voting. For topic relevance they build profiles for each user and each URL separately, modeling them as bag-of-words vectors. The similarity of users and links can then be computed by calculating the cosine similarity between vectors. Furthermore, they use the local graph as an indirect voting procedure, where the existence of the URL in

a message constitutes a vote from the author of the message for that URL. The work of Phelan et al. [160] also presents a system which can recommend URLs, but in this case they are augmenting the output of an RSS reader with information from Twitter. Their system, called Buzzer, ranks RSS stories based on the co-occurrence of popular terms within the user's feeds and Twitter messages. Similar work also includes Khabiri et al. [112] where the authors try to predict the popularity of new messages in Digg, as measured by up/down votes. In addition, Castillo et al. [38] try to predict the number of citations of academic papers given the information on both authors and the citation network.

Chapter 5

Personalized Information Filtering in Twitter

5.1 Introduction

In the previous chapter, we discussed how popular information can be identified and predict in Twitter. Merely recommending popular messages to all users, regardless of personal tastes, is not uncommon in online conversational media services. However, it is believed that more personalized information filtering can help users find relevant information and improve user experiences. This chapter and the following chapter dedicate to this idea.

In the current online social media ecosystem, users are able to connect and communicate with each other utilizing rich multimedia content, including text, images, video, and audio. These communication streams allow users to be informed instantly of the latest updates from their social connections. As a result, the aggregated social content has become a powerful tool for monitoring critical information in various situations, such as natural disasters and political upheavals.

Although services like Twitter and Facebook provide platforms for users to obtain fresh

information, two issues prevent them from being sufficiently relevant, causing deterioration of user experience and engagement. First of all, when facing a large amount of content from their social connections, users simply cannot consume it in an effective and efficient way, leading to the problem of *information overload*. On the other hand, information for a user is usually limited in scope to the user's social connections. Thus, it is difficult for a user to obtain information distributed outside of their circle, even though it might match their interests, leading to the problem of *information shortage*. Users may spend a significant amount of time filtering and searching for relevant information in social media. From the perspective of service providers, it is also very important to understand how users interact with the systems through a variety of actions such as re-posting (retweets), replying and commenting. It is also indispensable to track what in which users are interested, induced from the content requested and generated by them. Therefore, social media services can filter and recommend content to users based on their the history of previous interactions and interests.

The task of understanding users' behaviors and their interests has a number of challenges. First, although the number of items (updates, tweets, etc.) generated by users in such services may be huge, few of them impact a particular user, making the interaction data sparse. Second, new users and new content items flow into the system continuously. Thus, the "cold start" problem tends to be severe in these social platforms, compared to traditional information systems. In addition, a tremendous amount of content is rich yet noisy. Simple information retrieval or topical modeling techniques may not be sufficient to capture users' interests.

The problem tackled in this chapter has strong links to research in recommender systems and collaborative filtering. From the perspective of recommender systems, the task can be cast as building a list of relevant content items to users based their social connections and interests. Thus, many collaborative filtering techniques are applicable to

the task. However, on the other hand, as we mentioned, social content systems are much more dynamic than traditional recommender systems: many new items are pushed into the system every second. Therefore, recommender systems need to be adapted to this novel situation.

State-of-the-art collaborative filtering models are based on latent factor models (`LFM`), partially due to their superior performance in the Netflix competition (e.g., Koren [121]). However, the basic assumption for standard `LFM` is to exploit a user-item interaction matrix and cannot handle arbitrary features easily. Although some of newly proposed frameworks such as Agarwal and Chen [1] and Chen et al. [48], based on `LFM`, can consider features, fundamental modeling assumptions prevent them from handling high-order interaction data (e.g., tensors). In addition, current extensions to `LFM` such as Agarwal and Chen [2], Shan and Banerjee [178] and Wang and Blei [195] which incorporate rich text information are usually cumbersome, requiring complicated inference algorithms which cannot scale to large datasets. Moreover, researchers in collaborative filtering are realizing that pointwise-based measurement may no longer be appropriate and so a handful of ranking-based metrics are proposed. However, no work to date has compared them systematically on real world datasets.

In this chapter, we study the problem of modeling users' behaviors by focusing on one particular decisions, retweets, in Twitter and try to understand users' interests. Our method can be easily extended to model multiple types of users' decisions as well. We use a state-of-the-art recommendation model, Factorization Machines `FM` [165], to model user decisions and user-generated content simultaneously. Our contributions can be summarized as follows:

- We propose Co-Factorization Machines (`CoFM`), which deal with two (multiple) aspects of the dataset where each aspect is a separate `FM`. This type of model can easily predict user decisions while modeling user interests through content at the

same time.

- We apply Factorization Machines to text data with constraints. Thus, the resulting method can mimic state-of-the-art topic models and yet benefit from the efficiency of a simpler form of modeling.

- For user decision modeling, we compare a number of ranking-based loss functions. We introduce the newly proposed WARP loss Usunier et al. [190], which has been successfully applied in text and image retrieval tasks (e.g., Weston et al. [208] and Weston et al. [209]), into the context of recommendation.

- We apply our proposed methods to the problem of modeling personal decision making in Twitter and explore a wide range of features, revealing which types of features contribute to the predictive modeling and how content features help with the prediction.

We next review several directions of related work. In Section 5.2, we review `FM`: its basic settings and learning procedure. In Section 5.3, we formalize `CoFM` with different strategies of shared latent features. In Section 5.4, we compare and discuss a variety of loss functions. In Section 5.5, we describe features used in our model. In Section 5.6, we report the experiments with the discussions of datasets and baselines used. We conclude the chapter in Section 5.7.

5.2 Modeling Twitter by Factorization Machines

In this section, we first review the basic settings of `FM` with a discussion of how it can be used for different types of responses. Then, we detail how `FM` can fit into the task of modeling Twitter data, which is a predictive modeling for user responses and a topic coding model for content.

5.2.1 Factorization Machines

Here we briefly review `FM`, a state-of-the-art framework for latent factor models with rich features. For a detailed description, please refer to Rendle [165]. We start from a design matrix $\mathbf{X} \in \mathbb{R}^{D \times P}$, where the ith row $\mathbf{x}_i \in \mathbb{R}^P$ denotes one data instance with P real-valued variables and where y_i is a response for the data instance. We use s_i to represent our estimation of y_i based on $\mathbf{x}_i$. In many tasks, the goal is to make the discrepancy between s_i and y_i as small as possible. The factorization machine (FM) model of order $d = 2$ for $f(y_i \,|\, \mathbf{x}_i, \boldsymbol{\Theta}) = s_i$ can be defined as:

$$s_i = \beta_0 + \sum_{j=1}^{P} \boldsymbol{\beta}_j \mathbf{x}_j + \sum_{j=1}^{P} \sum_{j'=j+1}^{P} \mathbf{x}_{i,j} \mathbf{x}_{i,j'} \sum_{k=1}^{K} \boldsymbol{\theta}_{j,k} \boldsymbol{\theta}_{j',k} \tag{5.1}$$

where k is the dimensionality of the factorization and the model parameters $\boldsymbol{\Theta} = \{\beta_0, \boldsymbol{\beta}, \boldsymbol{\theta}\}$ are: $\beta_0 \in \mathbb{R}$, $\boldsymbol{\beta} \in \mathbb{R}^P$ and $\boldsymbol{\theta} \in \mathbb{R}^{P \times K}$. The form of FM (Equation 5.1) is very general and can be used in many applications. In order to cope with different tasks, we can define an exponential family distribution over $P(y_i \,|\, s_i)$ similar to that utilized in Lee et al. [127], as $P(y_i \,|\, s_i) = h(y_i, \boldsymbol{\Phi}) \exp\Big[\eta(\lambda; s_i)^T T(y_i) - A\Big(\eta(\lambda; s_i)\Big)\Big]$ where λ is the **natural** parameter of the family. For instance:

- **Gaussian response**: For normal rating prediction problems, y_i can be treated as a real-valued response drawn from a Gaussian distribution. Thus, we scale s_i with a known variance. $\eta = s_i/\sigma^2$, $A = s_i^2/2\sigma^2 = \sigma^2\eta^2/2$ and $h = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2} y_i^2)$ where σ^2 is the variance. Thus, the expectation $\mathbb{E}[T(y)] = s$.

- **Poisson response**: For word counts, y_i can be treated as the indicator of word index, drawn from a Poisson distribution. Thus, $\eta = s_i$, $A = \exp(s_i)$ and $h = \frac{1}{y_i!}$. The expectation $\mathbb{E}[T(y)] = \exp(s) = \lambda$.

- **Binary (Bernoulli) response**: For binary decisions, y_i is usually treated as 0 or

1. Thus, $\eta = s_i$, $A = \log(1 + \exp(s_i))$. The expectation $\mathbb{E}[T(y)] = \frac{\exp(s)}{1+\exp(s)}$.

One important aspect of `FM` is that the model can mimic the structure of many state-of-the-art models like matrix factorization, pairwise interaction tensor factorization, SVD++ and neighborhood models in one unified framework, as demonstrated by Rendle [165].

`FM` can be learned in several ways, including maximum a posteriori (MAP) estimation and Bayesian inference. In this chapter, we stick to MAP learning, which is to maximize the log likelihood through stochastic gradient descent (SGD) or alternating least squares. An equivalent view is to minimize a loss between the reconstructed response $f(y_i \mid \mathbf{x}_i, \boldsymbol{\Theta})$ and the true response y_i as $\arg\min_{\boldsymbol{\Theta}} \sum_i l(f(y_i \mid \mathbf{x}_i, \boldsymbol{\Theta}), y_i) + \mathcal{R}(\boldsymbol{\Theta})$ where l is a certain loss function and $\mathcal{R}$ is a regularization term for $\boldsymbol{\Theta}$.

5.2.2 Modeling User Decisions and Content

We focus on extending `FM` to model Twitter data. We have two goals to achieve for understanding and modeling user behaviors in Twitter. First, we wish to uncover what kind of items with which users interact through various actions (e.g., retweets, replies and favorites) and what features contribute to the mechanism that causes certain pieces of information to be shared across social connections. Second, content is of great importance in Twitter and thus it is vital to discover topics in which users are interested and how these topics influence users' decisions. As mentioned earlier, we wish to predict users' decisions as accurately as possible while discerning topics from the huge amount of user-generated content at the same time.

For the first task, we focus on a binary response y_d for each tweet d: whether the tweet will be retweeted by a target user $u^{(t)}$. We use $u^{(a)}$ denote the author of the current tweet d. For each tweet, we use $\mathbf{x}_d$ to represent a list of explicit features for tweet d, which might include features about $u^{(t)}$ and $u^{(a)}$, and $\boldsymbol{\theta}_d$ to represent a list of latent features (factors) discovered through modeling. For the purpose of discussion, we compose a simple feature

vector consisting of one categorical feature to indicate its author and one categorical feature to indicate the index of the tweet. Following the definition of `FM` (Equation 5.1), we can have an estimation of y_d based on $\mathbf{x}_d$ and $\boldsymbol{\theta}_d$ as $f(s_d \,|\, \mathbf{x}_d, \boldsymbol{\theta}_d)$:

$$s_d = \beta_0 + \boldsymbol{\beta}_u + \boldsymbol{\beta}_d + < \boldsymbol{\theta}_u, \boldsymbol{\theta}_d > \tag{5.2}$$

This is the exact form of traditional matrix factorization for collaborative filtering where the $\boldsymbol{\theta}$ are treated as latent factors for users and items (tweets). Since y_d is a binary response, we can also have a binary version as $y_d \sim \text{Bernoulli}\Big(\delta(s_d)\Big)$ where δ is the sigmoid function. For user decisions $\mathbf{y}$, one natural criteria for a reasonable model is to predict them as accurately as possible by specifying some error based loss functions (e.g., squared error loss). Alternatively, we can provide a ranked list of items for each user and measure how well these ranked lists approximate user actions. The latter view goes beyond pointwise evaluation and learning process for recommendation systems and has been studied extensively recently (e.g., Koren [121] and Yang et al. [214]). Later, we will further explore this idea.

For the second task, we will model terms in each tweet d. We denote s_{dv} to represent our estimation of the raw word count of term v in tweet d, w_{dv}, which is the response in this task. The vector $\mathbf{x}_{di}$ is used to represent features of term i in tweet d. In order to explain things more clearly, we change $\boldsymbol{\beta}$ for s_{dv} to $\boldsymbol{\alpha}$ and latent factors to $\boldsymbol{\phi}$. We use two simple indicator features here. More specifically, for each s_{dv}, we associate one categorical feature to indicate the term index and another one to indicate the tweet index. Following a similar argument, s_{dv} is a function of all features $f(s_{dv} \,|\, \mathbf{x}_{dv}, \boldsymbol{\phi}_{dv})$:

$$s_{dv} = \alpha_0 + \boldsymbol{\alpha}_d + \boldsymbol{\alpha}_v + < \boldsymbol{\phi}_v, \boldsymbol{\phi}_d > \tag{5.3}$$

The inner product between $\boldsymbol{\phi}_v$ and $\boldsymbol{\phi}_d$ is of interest. For $\boldsymbol{\phi}_d$, it is a K-dimensional vector

and it can be treated as code for tweet d, playing a similar role as $P(z \mid \theta)$ in traditional topic models like probabilistic latent semantic analysis (`PLSA`) or latent Dirichlet allocation (`LDA`). The only difference is that $\boldsymbol{\phi}_d$ is not constrained to rest on the simplex. For $\boldsymbol{\phi}_v$, it is a K-dimensional vector for term v and it can be treated as a $P(z \mid v)$. Using $\boldsymbol{\phi}_v$ for all terms, we construct the following matrix:

$$\mathbf{M} \in \mathbb{R}^{K \times V} \quad \text{where } \mathbf{M}_{\cdot,v} = \boldsymbol{\phi}_v$$

where each column of $\mathbf{M}$ is set to $\boldsymbol{\phi}_v$. Therefore, each row of $\mathbf{M}$ can be treated as $P(v \mid z)$ **without** normalization. In order to recover a similar modeling power from topic models, we add the following constraints to the parameters:

$$\sum_v \boldsymbol{\phi}_{kv} = 1 \text{ for all } k, \boldsymbol{\phi}_{kv} \geq 0 \text{ ; and, } \boldsymbol{\phi}_{dk} \geq 0 \text{ ;}$$

Thus, we have a normalized matrix $\mathbf{M}$, resembling a conventional topic matrix. We can further restrict all $\boldsymbol{\alpha}$ to be non-negative, resulting in a non-negative decomposition of the term matrix. In such a setting, the content modeling behaves like a simpler topic model than its more complicated counterparts. If we view the terms in a tweet as count data, we can use the Poisson distribution and thus have $w_{di} \sim \text{Poisson}\Big(\exp(s_{di})\Big)$. On the other hand, since terms are sparse in tweets where one term will most likely appear only once in a single tweet, an alternative parametrization is to use the Bernoulli distribution $w_{di} \sim \text{Bernoulli}\Big(\delta(s_{di})\Big)$. Note that other explicit features, which are not discussed here, can be incorporated into the model easily.

Thus, we can have use `FM` to model two tasks separately. However, it might be better if we can jointly model these two aspects of the data together. In this chapter, we propose several methods to simultaneously perform these two tasks.

5.3 Co-Factorization Machines

In many application scenarios, we may have multiple views. For instance, each tweet is associated with two types of important aspects to be modeled: 1) user action responses and 2) terms. In this subsection, we introduce Co-Factorization Machines (`CoFM`) to address the problem of modeling multiple aspects of data. Following the setting described in Equation 5.2 and Equation 5.3, we have two separate `FM`s to model two aspects of the **same** tweet where the two aspects are not linked together. Notice that we have learned two different latent representations of the same tweet: $\boldsymbol{\theta}_d$ and $\boldsymbol{\phi}_d$. Linking two factorization machines might be possible if these two latent representations of the tweet can be coupled in certain ways. Indeed, there exist several design choices to combine these two modeling processes. We present three paradigms below.

5.3.1 CoFM through shared features

One natural approach to link two latent representations of the **same** tweet is to treat one type of latent representation as a set of features and feed it into another modeling process. More specifically, we plug $\boldsymbol{\phi}_d$ into Equation 5.2 and have:

$$\begin{aligned} s_d = \beta_0 + \boldsymbol{\beta}_u + \boldsymbol{\beta}_d + &< \boldsymbol{\beta}_{\boldsymbol{\phi}_d}, \boldsymbol{\phi}_d > + < \boldsymbol{\theta}_u, \boldsymbol{\theta}_d > \\ &+ \sum_{k=1}^{K} \boldsymbol{\phi}_{d,k} < \boldsymbol{\theta}_u, \boldsymbol{\theta}_{\boldsymbol{\phi}_{d,k}} > + \sum_{k=1}^{K} \boldsymbol{\phi}_{d,k} < \boldsymbol{\theta}_d, \boldsymbol{\theta}_{\boldsymbol{\phi}_{d,k}} > \end{aligned} \tag{5.4}$$

Here, the third term $< \boldsymbol{\beta}_{\boldsymbol{\phi}_d}, \boldsymbol{\phi}_d >$ is a simple regression part by using latent factors obtained from content. The last two terms are of interest. Each latent factor in $\boldsymbol{\phi}_d$ is **re-weighted** by the interaction between its projection to the latent space of $\boldsymbol{\theta}$ and

corresponding user/tweet latent factors. In other words, the last two terms can be re-written as:

$$\sum_{k=1}^{K} \boldsymbol{\phi}_{d,k} < \boldsymbol{\theta}_u, \boldsymbol{\theta}_{\boldsymbol{\phi}_{d,k}} > + \sum_{k=1}^{K} \boldsymbol{\phi}_{d,k} < \boldsymbol{\theta}_d, \boldsymbol{\theta}_{\boldsymbol{\phi}_{d,k}} > =$$
$$< \boldsymbol{\phi}_d, \boldsymbol{\omega}_{u,\boldsymbol{\phi}} > + < \boldsymbol{\phi}_{di}, \boldsymbol{\omega}_{d,\boldsymbol{\phi}} >$$

where $\boldsymbol{\omega}$ are weights that each element is obtained from the interaction between the latent factors $\boldsymbol{\theta}$ and linear mapping vector $\boldsymbol{\theta}_{\boldsymbol{\phi}}$. This kind of mapping is similar in spirit that used by Gantner et al. [74]. The same process can be used for $w_{i,v}$ as well. If the shared feature mechanism is indeed a feature re-weighting scheme, another re-weighting approach might also be possible. Instead of using $\boldsymbol{\theta}_{\boldsymbol{\phi}}$ to map each dimension in $\boldsymbol{\phi}$ to $\boldsymbol{\theta}$, we treat $\boldsymbol{\phi}$ as the latent representation of a **missing** feature $\omega_{\boldsymbol{\phi}}$. The corresponding equation is:

$$\begin{aligned} s_d = \beta_0 + \boldsymbol{\beta}_u + \boldsymbol{\beta}_d + \boldsymbol{\beta}_{\omega_{\boldsymbol{\phi}}} \omega_{\boldsymbol{\phi},d} + < \boldsymbol{\theta}_u, \boldsymbol{\theta}_d > \\ + \omega_{\boldsymbol{\phi}} < \boldsymbol{\theta}_u, \boldsymbol{\phi}_d > + \omega_{\boldsymbol{\phi}} < \boldsymbol{\theta}_d, \boldsymbol{\phi}_d > \end{aligned} \tag{5.5}$$

Under this formalism, $\omega_{\boldsymbol{\phi}}$ is a missing feature and can be treated as weights for interactions between $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$. Comparing Equation 5.5 and Equation 5.4, we can see that the second formalism has fewer parameters to be estimated and more intuitive motivations. In this chapter, we use the second formalism.

5.3.2 CoFM through shared latent space

The methods introduced in the previous section assume that the latent representations obtained by different aspects of the model are different. A simpler approach is to assume that the latent factor of $\boldsymbol{\theta}_d$ is exactly the same as $\boldsymbol{\phi}_d$. Therefore, some parts of the latent factors of the same tweet are shared across different aspects. If we use $\boldsymbol{\eta}$ to indicate the

shared latent factors, the two aspects under this formalism are as follows:

$$
\begin{aligned}
s_d &= \beta_0 + \boldsymbol{\beta}_u + \boldsymbol{\beta}_d + < \boldsymbol{\theta}_u, \boldsymbol{\eta}_d > \\
s_{d,v} &= \alpha_0 + \boldsymbol{\alpha}_d + \boldsymbol{\alpha}_v + < \boldsymbol{\phi}_v, \boldsymbol{\eta}_d >
\end{aligned} \tag{5.6}
$$

This formalism shares the idea of matrix co-factorization used in relational learning scenarios.

This approach would be convenient when multiple factors will be shared. For instance, for each term, we can add one more categorical feature to indicate the author of the tweet and therefore obtain a latent representation of its author through content modeling:

$$
\begin{aligned}
s_d &= \beta_0 + \boldsymbol{\beta}_u + \boldsymbol{\beta}_d + < \boldsymbol{\eta}_u, \boldsymbol{\eta}_d > \\
s_{d,v} &= \alpha_0 + \boldsymbol{\alpha}_d + \boldsymbol{\alpha}_v + \boldsymbol{\alpha}_u + < \boldsymbol{\phi}_v, \boldsymbol{\eta}_d > + < \boldsymbol{\phi}_v, \boldsymbol{\eta}_u > \\
&\quad + < \boldsymbol{\eta}_u, \boldsymbol{\eta}_d >
\end{aligned}
$$

where the factors for user u and tweet d are shared in two aspects.

5.3.3 CoFM via latent space regularization

The discussions in Sections 5.3.1 and 5.3.2 represent two variations of how to work with two latent representations of the tweet. One can regularize two such representations such that they do not reside too far away from each other. A simple approach is to impose the following regularization on the model:

$$
\lambda_{\boldsymbol{\phi},\boldsymbol{\theta}} ||\boldsymbol{\phi}_d - \boldsymbol{\theta}_d||_F^2
$$

where $\lambda_{\boldsymbol{\phi},\boldsymbol{\theta}}$ is a regularization parameter. Under this assumption, we can also view that one latent factor is drawn from the multivariate normal distribution with the mean as

another latent factor:

$$\boldsymbol{\phi}_d \sim \text{MVN}(\boldsymbol{\theta}_d, \lambda_{\boldsymbol{\phi},\boldsymbol{\theta}}^{-1}\mathbf{I})$$

This will recover the formalism from Wang and Blei [195]. An additional possibility is suggested in Agarwal and Chen [3] where a "global" representation is assumed. The "local" representation is drawn from the "global" representation by a multivariate normal distribution. Thus, a third latent representation will be introduced if this approach is used.

5.4 Learning with CoFM

In this section we formalize the `FM` learning problem in an optimization framework. The discrepancy between the estimation s_i by `FM`/`CoFM` and the true value y_i can be measured by loss functions. Different choices of loss functions may lead to significant changes in performance as we will see in the experiments. Traditionally, pointwise error-based loss functions are widely used in latent factor models, which are widely used in recommender systems. Here, we discuss how different loss functions fit into the `FM`/`CoFM` framework and how the overall learning algorithm proceeds.

5.4.1 Optimization with content

For the task of modeling content in Twitter, two possible loss functions come from two assumptions (Poisson or Bernoulli distributions). We have the following optimization task:

$$\begin{aligned}
&\text{Opt(C)} = \arg\min_{\boldsymbol{\Phi},\boldsymbol{\alpha}} \sum_{d=1}^{D}\sum_{v=1}^{V} l_C(w_{dv}, f(s_{dv} \,|\, \mathbf{x}_{dv}, \boldsymbol{\alpha}, \boldsymbol{\Phi})) \\
&+ \sum_{j=1}^{P_1} \lambda_{\boldsymbol{\alpha},j} \boldsymbol{\alpha}_j^2 + \sum_{j=1}^{P_1} \lambda_{\boldsymbol{\phi},j} ||\boldsymbol{\phi}_j||_F^2 \\
&\text{s.t.:} \boldsymbol{\alpha} \geq 0,\ \boldsymbol{\phi}_{kv} \geq 0, \forall k, v;\ \boldsymbol{\phi}_k \in \mathcal{P}, \forall k \in K;\ \boldsymbol{\phi}_{dk} \geq 0, \forall d, k
\end{aligned} \tag{5.7}$$

where P_1 is the number of features used for each term v in each tweet d and $\mathcal{P}$ is a $(K-1)$-simplex. We consider the following loss functions for this task:

- Log Poisson loss: $l^{LP}(w_{dv}, s_{dv}) = -w_{dv} \log s_{dv} + s_{dv}$. Minimizing this loss is actually equivalent to minimizing an unnormalized KL-divergence between observed counts w_{dv} and their reconstructions s_{dv}.

- Logistic loss: $l^{LG}(w_{dv}, s_{dv}) = \log[1 + \exp(-w_{dv} s_{dv})]$. Minimizing this loss is essentially performing logistic regression. Here, we only consider on/off of a term v in tweet d, dismissing its possible multiple occurrences.

The optimization problem in Equation 5.7 can be efficiently solved according to two facts: 1) Due to the property of *multilinearilty* [165], the model is linear with respect to each model parameter when others are fixed and, 2) Proposition 1 in Zhu and Xing [230] states that the optimal value of a single parameter when other are fixed is the maximum between zero and the value obtained by a non-constrained version of the same problem. Also, efficient methods (e.g., Duchi et al. [64]) exist to project real-valued vectors onto the simplex. Therefore, this optimization problem is solvable.

5.4.2 Optimization with user decisions

For modeling user decisions, we also formalize the problem as an optimization problem as follows:

$$\begin{aligned}\text{Opt(U)} = \arg\min_{\boldsymbol{\Theta},\boldsymbol{\beta}} \sum_{d=1}^{D} l_U(y_d, f(s_d \,|\, \mathbf{x}_d, \boldsymbol{\beta}, \boldsymbol{\Theta})) \\ + \sum_{j=1}^{P_2} \lambda_{\boldsymbol{\beta},j} \boldsymbol{\beta}_j^2 + \sum_{j=1}^{P_2} \lambda_{\boldsymbol{\theta},j} ||\boldsymbol{\theta}_j||_F^2 \end{aligned} \tag{5.8}$$

where P_2 is the number of features used for each tweet d. Unlike content modeling, no constraints are put onto the parameter space. For pointwise loss functions, we consider:

- Squared error loss: $l^S(y_d, s_d) = (y_d - s_d)^2$, which is for regression problems with Gaussian responses.
- Logistic loss: l^{LG}, the same as the loss used in modeling content.
- Huber loss:

$$l^H(y_d, s_d) = \begin{cases} \frac{1}{2}\max(0, 1 - y_d s_d)^2 & \text{if } y_d s_d > 0 \\ \frac{1}{2} - y_d s_d & \text{otherwise} \end{cases}$$

  This is the one-sided variant of Huber's robust loss function. It is convex and continuously differentiable. The loss is mentioned in Yang et al. [213].

It has been demonstrated that pointwise loss functions may not be appropriate for recommender tasks when users choose items from a list of items prepared by the system [56]. In such cases, we wish to adopt more advanced loss functions which consider pairwise preferences. For each target user u, we can construct a set of tweets which are originated by other users and retweeted by u, denoted as $\mathcal{C}_u^+$. Note that these tweets could be originated by u's friends or any other users who are not connected to u. On the contrary, we denote all other tweets from u's friends which are not retweeted by u as $\mathcal{C}_u^-$. Therefore,

for each user u, there exists a huge set of tweets which are outside of u's network and are treated as unknown and not considered in the following loss functions. In this work, we focus on pairwise loss functions:

Margin ranking criterion (`MRC`):

$$l^M(x_1, x_2) = \sum_{x_1 \in \mathcal{C}_u^+} \sum_{x_2 \in \mathcal{C}_u^-} \max[0, 1 - f(x_1) + f(x_2)]$$

which considers all pairs of positive and negative labels, and assigns each a cost if the negative label is larger or within a "margin" of 1 from the positive label. Optimizing this loss is similar to optimizing the area under the curve of the receiver operating characteristic curve. That is, all pairwise violations are considered equally if they have the same margin violation, independent of their position in the list. For this reason the margin ranking loss might not optimize precision at k very accurately. This loss function is proposed in Herbrich et al. [84] and used in many IR tasks (e.g., [109, 16]).

Bayesian personalized ranking (`BPR`):

$$l^B(x_1, x_2) = \sum_{x_1 \in \mathcal{C}_u^+} \sum_{x_2 \in \mathcal{C}_u^-} -\log[\delta(f(x_1) - f(x_2))]$$

where δ is a sigmoid function. This loss is proposed in Rendle et al. [167] and has been used in tag recommendation (e.g., [168]) and yielded superior performance. This can be viewed as a smooth version of `MRC`.

Weighted Approximately Ranked Pairwise loss (`WARP`): This loss, proposed in Usunier et al. [190], has been successfully applied in image retrieval tasks [208] and IR tasks [209]. Here, we discuss its application in recommender systems. The idea of `WARP` is to focus more on the top of the ranked list where the top k positions are those we care about, comparing to `MRC` and `BPR` where no notion of ranked list is introduced. By

using the precision at k measure, one can weigh the pairwise violations depending on their position in the ranked list. `WARP` is defined as an error function as follows:

$$\text{err}_{\text{WARP}} = \sum_{\mathbf{x}_i \in \mathcal{C}_u^+} L[rank(f(\mathbf{x}_i))] \tag{5.9}$$

where $rank(f(\mathbf{x}_i))$ is the rank of a positive labeled instance $\mathbf{x}_i \in \mathcal{C}_u^+$ given by $rank(f(\mathbf{x}_i)) = \sum_{\mathbf{x}' \in \mathcal{C}_u^-} \mathbb{I}[f(\mathbf{x}') \geq f(\mathbf{x}_i)]$ where $\mathbb{I}(x)$ is the indicator function, and $L(\cdot)$ transforms this rank into a loss: $L(r) = \sum_{j=1}^{r} \tau_j$, with $\tau_1 \geq \tau_2 \geq \cdots \geq 0$. The idea of the $rank$ function is to compute the violations where negative instances are ranked higher than the positive ones and the L function is to transform violations into a loss. Different choices of τ define different importance of the relative position of the positive examples in the ranked list. In particular:

- For $\tau_i = 1$ for all i we have the same AUC optimization as margin ranking criterion.
- For $\tau_1 = 1$ and $\tau_{i>1} = 0$ the precision at 1 is optimized.
- For $\tau_{i \leq k} = 1$ and $\tau_{i>k} = 0$ the precision at k is optimized.
- For $\tau_i = 1/i$ a smooth weighting over positions is given, where most weight is given to the top position, with rapidly decaying weight for lower positions. This is useful when one wants to optimize precision at k for a variety of different values of k at once.

In this chapter, we use $\tau_i = 1/i$. It is difficult to directly optimize `WARP` due to the discrete nature of indicator functions. In addition, since the number of negative instances is significantly larger than positive instances, the $rank$ function is inefficient to be calculated. Before we tackle these issues, the form of Equation 5.9 can be readily re-written as (see

[208] for details):

$$\text{err}_{\text{WARP}} = \sum_{\mathbf{x}_i \in \mathcal{C}_u^+} \frac{L[rank(f(\mathbf{x}_i))] \sum_{\mathbf{x}' \in \mathcal{C}_u^-} \mathbb{I}[f(\mathbf{x}') \geq f(\mathbf{x}_i)]}{rank(f(\mathbf{x}_i))}$$

with the convention $0/0 = 0$ when the correct label y is top-ranked. We replace the indicator function by using the margin function $\max(0, 1 - f(\mathbf{x}_i) + f(\mathbf{x}'))$. In order to approximate the *rank* function, for a given positive instance, one draws negative instances until the one which violates the indicator function. Thus, we approximate $rank(f(\mathbf{x}_i))$ by using $\left\lfloor \frac{D^- - 1}{N} \right\rfloor$ where $\lfloor \cdot \rfloor$ is the floor function, D^- is the number of items in $\mathcal{C}_u^-$ and N is the number of trials of sampling until a violating pair is found. The approximation only requires local knowledge of negative instances, making it easily to be calculated per user for our case.

Competitive softmax loss (`SOFTMAX`):

$$P(y_i = 1 \,|\, \mathcal{C}_u) = \frac{\exp(f(s_i \,|\, \mathbf{x}_i))}{\sum_{\mathbf{x}_i} \exp(f(s_i \,|\, \mathbf{x}_i))} \text{ for all } \mathbf{x}_i \in \mathcal{C}_u \tag{5.10}$$

This loss is introduced by Yang et al. [214] and is motivated by the idea that users are presented a list of items and they choose items based on the "utility" they will receive if the item is chosen. Here, we assume that the utility for item i consists of two components $s_i + e_i$ where s_i encodes the intrinsic interest of the item to the target user and e_i is a stochastic error term reflecting the uncertainty and complexity of the choice process. We choose s_i to be the outcome from the predictive model (e.g., `FM`, `CoFM`). If the error term e_i is independently and identically distributed as a Weibull distribution, the probability item i is chosen is exactly as Equation 5.10, which is essentially a multinomial logic model.

Competitive hinge loss (`HINGE`): Following a similar assumption that a user chooses items based on their utilities, we can formalize the problem of distinguishing positive

instances from negative ones as a problem of classification. Therefore, the key idea of `HINGE` loss is that the utility difference between a positive item and negative items would be greater than **random errors**, namely:

$$\mathbb{I}(y_i == 1) f(s_i \,|\, \mathbf{x}_i) > \frac{1}{|\mathcal{C}_u^-|} \sum_{\mathbf{x}_i \in \mathcal{C}_u^-} \mathbb{I}(y_i == 0) f(s_i \,|\, \mathbf{x}_i)$$

With this spirit, a pairwise preference learning problem can be formalized as follows:

$$\begin{aligned} & l^H = \min \sum_{t=1}^{|\mathcal{C}_u^+|} \xi_t \\ & \text{s.t.: } f(s_t \,|\, \mathbf{x}_t) - \frac{1}{|\mathcal{C}_u^-|} \sum_{\mathbf{x}_i \in \mathcal{C}_u^-} f(s_i \,|\, \mathbf{x}_i) \geq 1 - \xi_t \,\text{and}\, \xi_t \geq 0, \\ & \forall \mathbf{x}_t \in \mathcal{C}_u^+ \end{aligned}$$

where ξ are introduced parameters to be optimized. This loss reflects the insight that user decisions are usually made by comparing alternatives and considering the differences in potential utilities. In other words, the marginal utility between user choice and the average of non-choices is maximized. This loss is also introduced in Yang et al. [214].

All of these ranking-based loss functions were proposed in different contexts and never compared in recommender systems. From the discussion above, it is clear that `WARP` is the only loss function considering the relative positions between positive instances. Meanwhile, both `SOFTMAX` and `HINGE` consider the set of positive and negative instances as a whole, rather than `MRC` and `BPR` only deal with pairwise preferences. In addition to what has been discussed here, other ranking based loss measures are also proposed. For instance, Koren and Sill proposed the OrdRec approach [122], which is based on a pointwise ordinal model. The idea of OrdRec is to predict a full probability distribution of the expected item ratings, rather than only a single score for an item. However, this is not desirable in

Algorithm 1: The sketch of the algorithm to optimize Equation 5.11. This is one iteration over the whole dataset.

for $u = 1$ **to** $|U|$ **do**
  Optimize Opt(U) for $\boldsymbol{\theta}_d$, $\boldsymbol{\theta}_u$ and $\boldsymbol{\beta}$:
    Perform stochastic gradient descent for pointwise loss functions or rank-based loss functions.
  Optimize Opt(C) for $\boldsymbol{\phi}_d$ and $\boldsymbol{\alpha}$:
    Perform stochastic gradient descent for log-Poisson loss or logistic loss
Optimize Opt(C) for M
  Perform gradient descent to obtain the current optimal value for the topic matrix

our setting in that we only care about relevant items ranked at the top. Thus, it is not necessary (and even impossible in practice) to predict a full distribution over all positions. Other researchers have tried to optimize NDCG directly. However, this is either done by approximation [205] or by a two-stage approach [17], which might be sub-optimal.

5.4.3 Summary

Putting things together, a `CoFM` framework for learning a model for user decisions and content understanding can be formalized as:

$$\text{Opt(CoFM)}: \ \text{Opt(U)} + \pi \text{Opt(C)} \tag{5.11}$$

where π is a parameter to balance two objective functions. By choosing different coupling strategies introduced in Section 5.3 and different loss functions, Opt(CoFM) can effectively perform predictive modeling and maximum likelihood estimation of content at the same time. We adopt a hybrid of SGD and coordinate descent to optimize Equation 5.11, which is sketched in Algorithm 1. We iterate the whole dataset by performing SGD for predictive modeling and content modeling while fixing the topic matrix. After one iteration, we optimize the topic matrix by gradient descent. Since we restrict $\mathbf{M} \in \mathcal{P}$, an

efficient method [64] can achieve this task.

5.5 Features

Here we discuss the features used in our models. These features utilize the content of tweets that users have generated. All of these features try to capture users' interests. Features are divided into five groups: 1) categorical features, 2) content profiles, 3) relevance scores, 4) latent topic model features, and 5) content meta features, where each group has multiple features.

Categorical Features: The key idea of `FM/CoFM` is to use both indicator features and explicit features together to obtain competitive performance in predictive tasks. For modeling user decisions, we use three categorical features: 1) target user id, 2) neighbor user id and 3) the tweet id. For content modeling, we use the term id and the tweet id as features.

Content Features: For "content profiles" we utilize features to characterize what users have posted and what their friends have posted. For each tweet i, let $\boldsymbol{w}_i$ be the term vector for this tweet. Let $u(i)$ be the author of the tweet i. For user u, we construct three user profiles as follows:

- **Content Profile**: Let $\text{CP}(u) = \sum_{i=1}^{D} \mathbb{I}[u(i) == u]\mathbf{w}_i$. This is essentially the concatenation of all term vectors generated by user u.
- **Neighborhood Profile**: Let $\text{NP}(u) = \sum_{u' \in N(u)} C(u')$ where $N(u)$ is a set of friend users of user u.
- **Retweet Profile**: Let $\text{RP}(u) = \sum_{i=1}^{D} \mathbb{I}[u(i) == u \wedge r(i) == 1]\mathbf{w}_i$ where $r(i)$ is a binary indicator for whether tweet i is a retweet or not.

These profile features will capture the interests of users at a fine-grained level. One drawback of these features is that they capture the long-term interests of users. For new tweets, they remain the same and would be less discriminative. Thus, we introduce the second group of features, characterizing how relevant a new tweet is against user profiles. Let $\mathcal{R}(\mathbf{w}_1, \mathbf{w}_2)$ be a relevance measurement between term vector $\mathbf{w}_1$ and $\mathbf{w}_2$. Thus, we have the following relevance scores:

- **Content Relevance**: $\mathcal{R}(\mathbf{w}_i, \mathrm{CP}(u))$, measuring the similarity of the incoming tweet to the user's content profile.
- **Neighborhood Relevance**: $\mathcal{R}(\mathbf{w}_i, \mathrm{NP}(u))$, measuring the similarity of the incoming tweet to the user's neighborhood profile.
- **Retweet Relevance**: $\mathcal{R}(\mathbf{w}_i, \mathrm{RP}(u))$, measuring the similarity of the incoming tweet to the user's retweet profile.

We use dot product as the relevance measure although many other IR relevance scores could also apply. Both "content profiles" and "relevance scores" utilize term level information to determine the features. In addition to these features directly related to the content generated by the users, other meta information also might be useful:

- **Length of Tweet**: Number of characters used in tweet i.
- **Hash Tag Count**: Number of hash tags used in tweet i.
- **Hash Tag History**: How many times the hash tag appears in $u(i)$'s retweets.
- **URL Count**: The number of URLs used in tweet i.
- **URL Domain History**: How many times the URL domain appears in $u(i)$'s retweets.

- **Retweet Count**: The number of times tweet i has been retweeted so far.

Local Graph & User Features: These features potentially characterize how popular and how well connected a user is. Intuitively, a popular user who has many friends and followers can be actively passing information by retweeting messages.

- **Mention Count**: The number of times user u is mentioned in other tweets.
- **Friend Count**: The number of friends user u follows.
- **Follower Count**: The number of followers user u has.
- **Status Count**: The number of tweets user u has published.
- **Account Age**: Number of years user u appeared on Twitter.

User Relationship Features: Relation features refer to those features which represent the relationship between a target user u_i and his/her friend u_j.

- **Co-Friends Score**: This feature estimates the similarity of friend sets of the target user u_i and his/her friend u_j.
- **Co-Follow Score**: This feature estimates the similarity of follower sets of the target user u_i and his/her friend u_j.
- **Mention Score**: The number of times u_i mentions u_j.
- **Retweet Score**: The number of times u_i retweets u_j.
- **Mutual Friend**: Whether u_i and u_j are mutual friends.

The similarity measure used is the Jaccard coefficient.

Temporal Features: We estimate user u's activity level at time t as $h_u(t)$, which is calculated by the average number of tweets he/she published in a periodical time slot, e.g.,

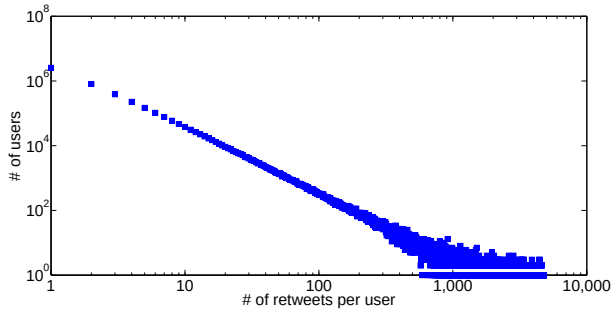

Figure 5.1: The sparsity of retweets per user.

every Monday. With the estimated response time Δt, the number of accumulated tweets can be written as: $r_u(\Delta t) = \sum_{j \in F(u)} \int_{t^w}^{t^w + \Delta t} h_j(t)\, dt$, as proposed by Peng et al. [157]. We calculate both activities using period of a day and a week.

In our case, all features are pre-calculated through a Hadoop cluster and can be processed efficiently.

5.6 Experiments

To prepare our dataset, we first monitored the Twitter public stream for one month in June 2012 and extracted users who post at least ten tweets including at least one retweet during this time period, resulting in 765,386 target users with approximately 11M tweets. For all these target users, we 1) crawled all their historical posts and 2) traced who they retweeted from and crawled all their posts as well. In this fashion, we obtained 4,327,816 neighbor users with 27M tweets, resulting in a dataset that is significantly larger than any previous work for similar tasks. For each target user u, we treat all tweets from his/her neighbor users as incoming tweets. If a tweet d from incoming tweets are retweeted by user u, d is treated as a positive instance, and negative otherwise. We plot the unnormalized distribution of number of retweets per user in Figure 5.1, demonstrating that a great

number of users only retweet a limited number of times while some users retweet thousands of times.

We adopt rank-based metrics to evaluate the effectiveness of different models. We borrow Mean Average Precision (`MAP`) from the IR community. We define "precision" at position k (Precision@k) of all incoming items for a particular user as (# of retweets in top positions)/k. Then an average measure across all top m positions (Average Precision) for user u is defined as $(\sum_{k=1}^{m} \text{Precision@k} \times l_k)/(\#$ of retweets for ranked list of user $u)$ where l_k is a binary indicator whether the position k has been retweeted or not and m is the total number of positions evaluated. Note that Average Precision is evaluated per user. We can average it across all users, resulting in the `MAP` measure, as used similarly in [47, 91].

In order to mimic a realistic evaluation environment, we adopt a time-based evaluation process, significantly differing from Chen et al. [47] where a **fixed** ratio of training vs. testing dataset is used. (It is not clear whether the ratio is kept according to the time order.) In addition, we do not use cross validation as it violates the time order of data, yielding unfair advantages to certain models. Here, for each user, we split all incoming tweets into n consecutive time periods with **equal** number of tweets in each time period. We train models on one time period and test them on the next. This is a balance between cross validation and online training and testing. In our experiments, we set $n = 5$.

We compare several aspects of proposed methods and other state-of-the-art baselines:

- **Matrix factorization (`MF`)**: Categorical features, the target user id and the tweet id are used to feed into `FM`, yielding a `MF` model with biases, which is a solid baseline in many collaborative filtering tasks.

- **Matrix factorization with attributes (`MFA`)**: In addition to the categorical features used in `MF`, we add explicit features into the model, which essentially mimic

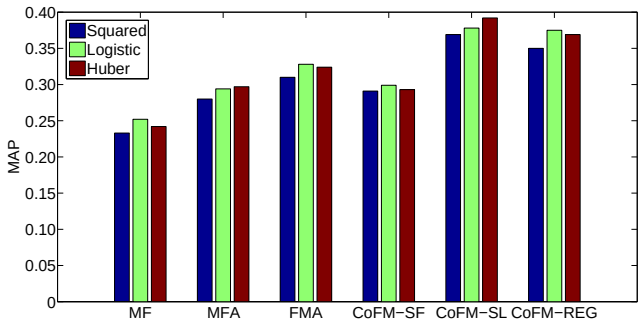

Figure 5.2: The comparison of pointwise loss functions.

the state-of-the-art latent factor models with features [1, 213, 45] as mentioned by Rendle [165].

- **Collaborative personalized tweet recommendation (`CPTR`)**: This is the model proposed in [47], which is a variation of `MFA` where the item factors for a tweet are decomposed into term factors and neighboring user factors. This is a state-of-the-art method for the task. We re-implement their approach under the framework of `FM`.

- **Factorization machines with attributes (`FMA`)**: In contrast to two categorical features, we add one more categorical feature, the neighbor user id, into the model, resulting in a pairwise tensor factorization model with "target user-item-neighbor user" interactions.

- **`CoFM` with shared features (`CoFM-SF`)**: This is the model introduced in Section 5.3.1 with the same indicator features as `FMA` for user decisions. We use term id and tweet id as two categorical features for content modeling. The latent factors for the tweet are shared through the tweet id. Thus, we have a pairwise tensor interaction model for user decisions and a topic coding model for content.

- **`CoFM` with shared latent spaces (`CoFM-SL`)**: This is the model introduced in Section 5.3.2. The feature setting is the same as `CoFM-SF`.

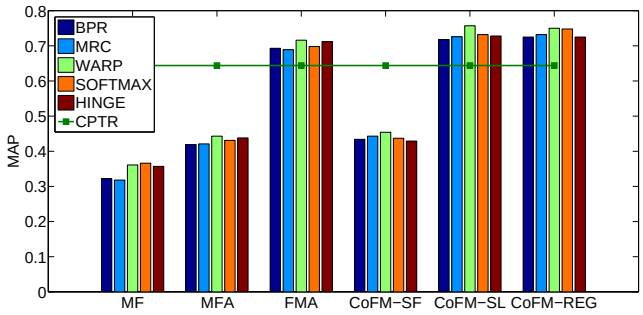

Figure 5.3: The comparison of ranking-based loss functions.

- **`CoFM` with latent space regularizations (`CoFM-REG`)**: This is the model introduced in Section 5.3.3. The feature setting is the same as `CoFM-SF`.

For the sake of simplicity, we fix all regularization parameters to 1 and tune π, the balance parameter of predictive modeling and content modeling in Equation 5.11. We report the performance of $\pi = 0.3$, which is the best in our experiments. We also tune K, the dimensionality of latent factors in `FM`/`CoFM`, from 10 to 250. We report the performance of $K = 150$, which is the best in the experiments. For `CPTR`, we fix $K = 200$, which is used in [47]. For content modeling, we do not observe significant differences between using log-Poisson loss and logistic loss. Thus, for generality, we report the results based on log-Poisson loss.

Predictive Results: We compare the predictive power of different models. First, we demonstrate how loss functions affect performance, starting from pointwise loss functions. For `MF`, `MFA` and `FMA`, we compare using three pointwise loss functions: squared error loss, logistic loss and Huber loss. For `CoFM-SF`, `CoFM-SL` and `CoFM-REG`, we use the same three loss functions for predictive modeling while fixing log-Poisson loss for content modeling. Since `CPTR` is fixed to pairwise learning, we exclude it from the experiment. The result is shown in Figure 5.2. The first observation is that the performance of `MF` which only uses user-item interactions is significantly worse than the ones utilizing explicit features.

The second observation is that logistic loss and Huber loss is consistently better than squared error loss. This might be due to the reason that we only have binary responses (retweets), similar to what is reported in Yang et al. [213]. The third observation is that `CoFM-SL` and `CoFM-REG` are noticeably better than all other methods. This validates our discussion in Section 5.3 that these two paradigms can be viewed as variants of many successful co-factorization models where the predictive aspect can benefit from the content modeling aspect. On the other hand, `CoFM-SF` performs poorly and even cannot match the performance of `FMA`. We conjecture that this is because the data is too sparse such that additional parameters induced by `CoFM-SF` cannot be effectively learned. We also observe that `FMA` performs better than `MFA`, indicating that ternary interactions "target user-item-neighbor user" can indeed capture the dynamics between users on Twitter, compared to "target user-item" binary interactions.

In addition to pointwise loss functions, we also compare performance of different models with ranking-based loss functions, as shown in Figure 5.3. The green line in the bar chart is the performance of `CPTR` since it is trained with `BPR`. Comparing the results to pointwise loss functions, the overall performance is significant higher, indicating that ranking-based loss functions are indeed better for the task. Also, the discrepancy between different models becomes larger, compared to pointwise loss functions. For instance, `FMA`, `CoFM-SL` and `CoFM-REG` are much better than the others, where all three are above `CPTR` significantly. In addition, `CoFM-SL` and `CoFM-REG` are consistently $3\% - 4\%$ better (depending on the specific ranking-based loss function) than `FMA` in absolute MAP scores across 5 split of data. Overall, `WARP` achieves competitive performance consistently for all models.

Content Modeling: We explore how topics are learned through the modeling. From the formalism in Section 5.4.1, the matrix $\mathbf{M}$ can be interpreted as a topic matrix as in standard PLSA/LDA. Thus, we can describe topics as in other topic models by ranking terms in probabilities. This is superior to `CPTR` [47] where term factors are not in the

Table 5.1: Examples of topics produced by CoFM.

Entertainment
album music lady artist video listen itunes apple produced movies #bieber bieber new songs
Finance
percent billion bank financial debt banks euro crisis rates greece bailout spain economy
Politics
party election budget tax president million obama money pay bill federal increase cuts

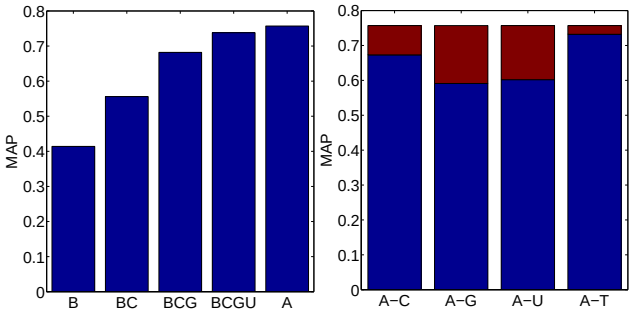

Figure 5.4: The impact of different groups of features.

simplex. We show some example topics in Table 5.1. The terms are top ranked terms in each topic and the topic names shown in bold are given by the authors. We can see that these topics are easily recognized and have the benefit of normal topic models while we do not have cumbersome Bayesian style formalism and expensive inference algorithms in the model. Note, however, that content modeling is not only for explanatory analysis—it is indeed helpful for prediction tasks. From Figures 5.2 and 5.3, we can see that `CoFM` which utilizes content modeling has better performance in general, and especially for `CoFM-SL` and `CoFM-REG` which can outperform state-of-the-art methods significantly.

Feature Analysis: We study how different types of features contribute to the predictive power of the model. Instead of using methods like χ^2 to calculate the correlation

between feature values with respect to classification labels, we adopt the following two straightforward methods. First, we start from a base model `CoFM-SL` using `WARP` without any explicit features, which is the best model from previous experiments, and then add one group of features consecutively. This method, denoted as "add on", shows the contribution of each group of features as it is added into the model. The second method, denoted as "take out", starts with a complete model and removes one group of features to see how performance drops accordingly. The results for "add on" and "take out" are shown in Figure 5.4. The effect of "add on" is shown on the left and the effect of "take out" is on the right. For both figures, "A", "B", "C", "G", "U" and "T" stand for "All", "Base model", "Content feature", "Graph feature", "User feature" and "Temporal feature" respectively. For "add on", it is clear that each group of features contributes to the final performance of the model and "Temporal" features have the least marginal gain. The most gains come from "Content" features and "Local Graph" features. This observation is consistent with [92]. For "take out", the red square in each bar in the figure represents the performance deduction for the corresponding feature group. Again, "Temporal" features have the least impact on performance while removal of "Local Graph" features hurts performance much more than that of "Content" features. From both "add on" and "take out", it seems that "Local Graph" plays an important role in the performance, followed by "Content" features. This suggests that social connections are important in determining retweets as well as content factors.

5.7 Summary

Users of social media services are often simultaneously overwhelmed with the amount of information delivered via their social connections and miss out on content that they might

have liked to see. Both issues serve as difficulties to the users and drawbacks to the services. These services can benefit from understanding user interests and how they interact with the service, potentially predicting their behaviors in the future. We propose Co-Factorization Machines (`CoFM`) to address the problem of simultaneously predicting user decisions and modeling content in social media by analyzing rich information gathered from Twitter. The task differs from conventional recommender systems as the cold-start problem is ubiquitous, and rich features, including textual content, need to be considered. Additionally, we discuss and compare ranking-based loss functions in the context of recommender systems, shedding light on how they vary from each other and perform in real tasks, providing the first work in this direction. We explore a large number of features and conduct experiments on a real-world dataset, concluding that `CoFM` with ranking-based loss functions is superior to state-of-the-art methods and yields interpretable latent factors.

5.8 Bibliographic Notes

In this section, we review three lines of relevant research work: 1) collaborative filtering and ranking, 2) collaborative filtering with content integration, and 3) Twitter user and content modeling. We link them with our tasks and discuss the novelty of our work as well.

Recommender systems which utilize `LFM` have gained significant attention because they were used by the winning team of the Netflix Prize. However, simple `LFM` cannot easily be coupled with additional information (features) in other recommendation scenarios. Recently, researchers have explored how traditional `LFM` can be enhanced by exploiting rich features generated by users. Three main paradigms are proposed for this purpose. The first paradigm is "Regression Based Factor Models" (`RBFM`) and its extensions, proposed by Agarwal, Chen and colleagues [1, 2, 3, 225], which have been successfully used in a variety

of recommendation scenarios, such as social networks [213, 214], professional networks and content recommendation. The basic idea behind RBFM is to replace zero-mean Gaussian distributions usually used in a simple LFM with regression-based means. Thus, RBFM adds another layer of regression on top of LFM, which can incorporate different types of features effectively. However, RBFM can only handle 2-order data interactions and thus high-order data structures (e.g., tensors) cannot be modeled. In addition, the proposed method, training with Monte Carlo EM, is inefficient and cannot scale to large scale datasets easily. The second paradigm is called "Factorization Machines" (FM), proposed by Rendle et al. [165]. FM can handle, in theory, arbitrary orders of interactions between variables and naturally deal with features. However, the existing formalism of FM has not been explored with topical modeling of content and pairwise preferences learning has not been discussed in the context of FM. The last paradigm is called "Feature-based matrix factorization" (FBMF), proposed by Chen et al. [45], which is to combine LFM with linear regression. As noted by Rendle [165], similar to RBFM, FBMF cannot handle high-order interactions either. Additionally, FBMF can be viewed as a special case of FM.

Traditional collaborative filtering methods are trained and evaluated on pointwise-based measures, such as Root-Mean-Square-Error (RMSE) and Mean-Square-Error (MSE), essentially measuring how accurate each single prediction is, regardless of how items are presented to users. Although the use of RMSE gained popularity through the Netflix prize competition, ranking-based measures might be more appropriate than pointwise-based measures since users are presented a ranked list of items calculated by recommendation systems. Thus, it would be more natural to optimize the ranked list directly. Some recent advances in recommender systems lie in this direction. For instance, Weimer et al. [205] extended the maximum margin matrix factorization method (MMMF) by optimizing a surrogate loss function approximating the NDCG ranking measure, a ranking-based metric commonly used in the Information Retrieval (IR) community. However, the method

proposed in the paper is complicated and arguably hard to scale to large datasets. A margin ranking criterion, an ordinal loss, is introduced from the field of IR to collaborative filtering by Weimer et al. [206] in the context of `MMMF`, which is a direct extension of the hinge loss for Support Vector Machines. This max-margin loss essentially minimizes AUC, which is the area under the ROC curve [16, 208]. A smooth version of hinge loss which is also to minimize AUC, called Bayesian personalized ranking, is proposed by Rendle et al. [167] and has yielded superior performance in tag recommendation. Recently, Balakrishnan and Chopra [17] proposed a two-stage procedure for collaborative ranking. Their proposal is to first train a matrix factorization model for users and items and treat latent factors as features to feed into a standard learning-to-rank framework. Koren and Sill [122] proposed a method to embed ordinal regression into matrix factorization by predicting a full distribution over all ranks. An interesting point is that this method is indeed a pointwise method. With a slightly different approach, Yang et al. [214] studied user behaviors when browsing a list of items. The proposed framework includes two loss functions that are comparable to multinomial logic model and a margin ranking criterion but have more intuitive explanations. Though different ranking-based loss are proposed, they are never compared.

Some recent developments in collaborative filtering have demonstrated the power to integrate rich content from articles and scientific papers with user decisions to provide better recommendation results. For instance, Agarwal and Chen [2] extended `RBFM` with a latent Dirichlet allocation prior for latent factor models. A similar approach was investigated by Shan and Banerjee [178] as well. Wang and Blei [195] proposed a method to combine matrix factorization with probabilistic topic modeling for recommending scientific papers. The method cannot easily leverage additional features. One significant advantage of these joint modeling methods is that latent factors obtained through content modeling can reveal interpretable meanings to latent spaces and thus provide a unique

way to uncover some hidden structures of the data. However, all these methods require complicated inference algorithms which are not easily to scale to large datasets.

Recommender systems have been built specifically for Twitter. For instance, Kim and Shim [115] argued that Twitter does not offer to find the most interesting tweet messages for users. The authors proposed a probabilistic model derived from Probabilistic Latent Semantic Analysis (PLSA) for collaborative filtering to recommend potential followers to users in Twitter. The method does not consider any explicit features at all. Due to the fact that Twitter users form a information network, researchers have tried to use undirected graphical models to model such networks. For example, Yang et al. [215] used a factor graph to model the spread of tweets. Lin et al. [132] proposed a generic joint inference framework for topic diffusion and evolution in social network communities based on Gaussian Random Fields, which also cannot integrate with rich features. Similarly, Peng et al. [157] proposed a method based on Conditional Random Fields (CRF) to predict how likely a tweet will be retweeted by a user. The proposed method suffered from the difficulty to efficiently perform inference on graph-like CRFs. Duan et al. [63] studied how learning to rank approaches can be used in ranking tweets. They explored a number of features and used 20 query terms as input to train a RankSVM as the model. In the present work, we do not have explicit queries while modeling user decisions and user-generated content. Hong et al. [92] and Uysal and Croft [191] trained classifiers to predict whether a tweet will be retweeted. However, the classifiers they trained are universal for all users and hence cannot provide personalized results. Recently, Chen et al. [47] utilized `FBMF` with a wide range of features to recommend tweets for users on Twitter. However, the proposed method cannot provide much insight on how content contributes to users' decisions and only one type of ranking loss function is used without comparisons. In all, these methods either do not handle arbitrary features or do not obtain summarized content (topics) from Twitter messages, preventing us from further understanding how

users' decisions are made. In this work, we will perform these two tasks simultaneously.

Chapter 6

Information Filtering in Professional Social Streams

In the current Web ecosystem, social media services are ubiquitous. With the rise of websites like Facebook, Twitter, and LinkedIn, millions of users are connecting and communicating with each other over rich multimedia content, including text, images, video, and audio. The content forms streams of social updates, which allow users to get instantly informed on the latest news. When aggregated, the social update streams become a powerful tool for monitoring the geopolitical situation in various regions. For instance, social update streams have been heavily used to disseminate information during the "Arab spring". In addition, social update services become the ultimate platform for collecting information in natural disasters, such as earthquakes and tsunamis [145].

Although social update streams provide a unique opportunity for users to obtain fresh information, users commonly acknowledge two issues that prevent current social streams from being sufficiently relevant, which causes deterioration of user experience and engagement. First of all, while facing a large number of updates from their social connections, users simply cannot consume them in an effective and efficient way. This is known as

the problem of *information overload* (see, e.g., [29, 92]). Furthermore, social updates for a user are usually limited in scope to their social circle, induced from their connections. Thus, it is very difficult for a user to obtain information distributed outside of their circle, even though it might match their interests. In order to obtain relevant information, users spend long hours searching social media (see, e.g., [21, 146]). We call this problem *information shortage*. To address both these problems, social media monitoring systems are being built, which filter and recommend social updates to users based on numerous signals. This area has recently attracted close attention of academic and industrial research communities.

The task of filtering and recommending social updates can be approached from various perspectives. From the Information Retrieval (IR) perspective, constructing personalized social streams can be cast into the classic ranking problem: the task is to rank social updates by descending order of user interest. It may be true that some existing IR techniques could be potentially applied to social stream ranking. However, user's interests in social streams are not represented in terms of a search query. Instead, queries are implicit and have to be inferred. The absence of a search query distinguishes the social stream ranking problem from many classic IR tasks. In addition, social information needs are more diversified compared to traditional IR scenarios. Although traditional IR tools do not appear to be directly applicable to ranking in social streams, some of recently developed *learning to rank* approaches are very appealing to be used in the new setting.

From the perspective of Recommender Systems (RecSys), building a list of relevant social updates can be viewed as recommending relevant items to users. Thus, many collaborative filtering techniques are applicable to the task of social stream ranking. However, social stream systems are much more dynamic than traditional recommender systems: many new updates can be pushed into the system every second. Therefore, the cold start problem becomes even more severe in social stream systems. The traditional collaborative

filtering paradigm needs to be adjusted to ranking social streams.

Surprisingly, little prior research work has been done to tackle the problem of social stream ranking from the point of view of building an effective system. This is partially due to the fact that no real-world dataset of social updates is openly available to the research community – due to obvious reasons related to user privacy. Most commercial ranking algorithms (e.g., the one used by Facebook) are proprietary. Indeed, a successful content ranking system on social streams will not only provide more relevant information to users and improve user engagement, but also shed the light on patterns of user behavior and social trends, which might be strong signals for behavioral targeting in computational advertising – the driving power of most Web 2.0 venues.

In this chapter, we start an open discussion on how to build effective systems for social stream ranking (SSR). To the best of our knowledge, we are the first group of researchers to elaborate technical details for such a system. More specifically, we address the problem as an intersection of learning to rank, collaborative filtering and clickthrough modeling, while leveraging ideas from information retrieval and recommender systems. Our contributions are three-fold:

1. Analyze social streams (based on LinkedIn data) and provide some insights on users' behavior;

2. Propose a novel probabilistic latent factor model with regressions on explicit features; integrate the idea of learning to rank and collaborative filtering

3. Demonstrate the superior performance of our model by comparing with several non-trivial real-world baselines.

The rest of the chapter is organized as follows. In Section 6.5, we compare SSR with other related research areas. As we will point out, SSR is a unique setup to which existing techniques cannot be applied directly. In Section 6.1, we review the problem of social

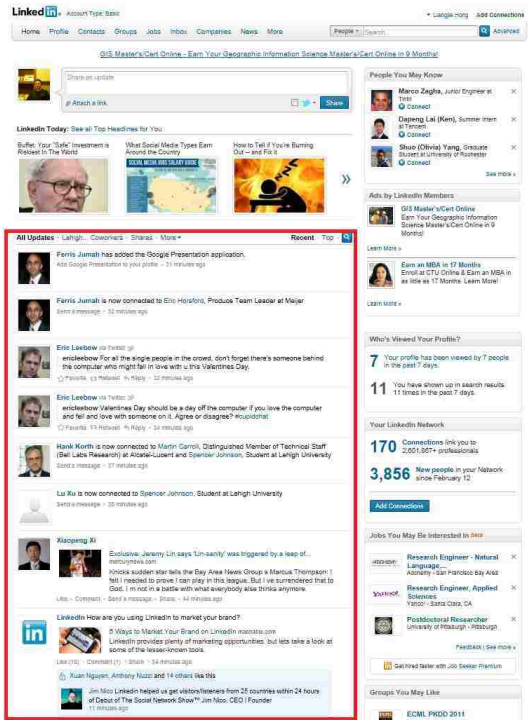

Figure 6.1: A typical example of LinkedIn homepage.

stream ranking in the context of LinkedIn. In Section 6.2, we introduce our ranking model step by step and compare it with some existing IR and collaborative filtering techniques. In Section 6.3, we evaluate our model against a number of non-trivial baselines. We conclude our paper in Section 6.4.

6.1 Overview of LinkedIn

Founded in December 2002 and launched in May 2003, LinkedIn[1] is primarily used for online professional networking. As of March 2012, LinkedIn has more than 160 million registered users in more than 200 countries and territories. On LinkedIn, user profiles play a central role for establishing professional existence and personal brand. Users can

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

update their professional profiles to include a spectrum of content types (e.g., position descriptions, publications, patents, open source projects, skills, etc.). In addition, LinkedIn offers collaborative platforms to help users consume relevant news stories (e.g., LinkedIn Today[2]), seek answers to questions on professional issues (e.g., LinkedIn Groups[3]), and share useful content (e.g., LinkedIn Signal[4]). On the left-hand side of LinkedIn's homepage, a typical user will see a list of content items that come from their professional connections. This update stream consists of a wide range of types of updates including changes on their profiles (e.g., changes in their employment), shares of information (e.g., news articles, blog posts), and Twitter updates. These updates compose a social stream for the user. A snapshot of a user's homepage is shown in Figure 6.1 where the social stream of a user is highlighted by a red rectangle, shown on the left hand side of the screen.

As we have discussed in previous sections, delivering truly relevant social updates to users is a very difficult task. Information overload is certainly a serious problem for users who have hundreds of connections. In contrast, for newly joined users who do not have a sufficient number of connections, a system could recommend potentially useful updates to make the user adapt to the service more smoothly and quickly.

From the LinkedIn point of view, it is important to attract users to consume their social content and interact with their social streams as it is a clear indicator of a healthy engagement pattern. A steady, but badly delivered social stream may distract the user and make them lose interest in the service. A weak social stream for new users may make them question the benefits of the service. On the other hand, if a user interacts with the social stream frequently, clicking "Like" buttons or making comments to others' updates, the user is likely to become more and more engaged.

[2]https://www.linkedin.com/today/
[3]https://www.linkedin.com/myGroups
[4]https://www.linkedin.com/signal/

6.2 Social Stream Ranking

In this section, we will discuss the ingredients of our proposed model step by step, from a simple linear model to a much more involved tensor factorization based latent factor model. We start our discussion from why the problem of SSR cannot simply be treated as a rating prediction problem, which is a classic setting in RecSys.

6.2.1 Evaluation Metrics

In traditional RecSys settings where the entities are users and items (e.g., the famous Netflix competition[5]) forming a matrix of users by items, the main goal is to predict or recover missing values in this matrix – if we treat existing ratings as observed values and non-existing ratings as missing values of the matrix. The performance of recommender systems is evaluated by how accurately a system can predict these values. For instance, in the Netflix Prize, Rooted Mean Squared Error (RMSE) is used to measure the accuracy of rating predictions. RMSE is defined as $\sqrt{\frac{1}{N}\sum_i^n (x_i - \hat{x}_i)^2}$ where x_i is the ground-truth rating for rating i, $\hat{x}_i$ is the predicted value and N is the total number of ratings to be tested. Although it might be an appropriate evaluation metric for movie recommendation tasks where multiple levels of ratings are available, two crucial issues will arise while using it as an evaluation metric to SSR.

First of all, as we see in Figure 6.1, the final presentation of a social stream is a list of items shown on the computer screen. Due to the limited space on the screen, users can only see a portion of the list which usually only consists of a handful of updates. Although users can always scroll down the list and even go to additional pages, not all of them do that in practice. Thus, even if the accuracy of predictions is important, we certainly wish to have higher-accuracy items on the top of the list rather than to have the whole list slightly more accurate. This ordering information cannot be easily captured

[5] http://www.netflixprize.com/

in accuracy-based metrics, like RMSE. In addition to the reason that accuracy-based metrics may not be appropriate, in the practical sense, a system optimizing accuracy might fail to produce reasonable results. As we will see, the users rarely interact with the *majority* of the updates. In other words, users are only interested in a small number of items. An accuracy-optimized system may overfit non-interacted items and yield good performance overall but might not match the small portion of clicked updates. Indeed, this drawback of accuracy-based metrics is also discussed in the collaborative filtering literature (e.g., [167, 122, 213]).

Based on this discussion, we adopt rank-based metrics to evaluate the effectiveness of SSR systems. Rank-based metrics are widely used in the IR community. In this paper, we borrow Mean Average Precision from the traditional IR. We define the "precision" at position k (Precision@k) of a social stream as (# of clicks in top positions)/k. Then, an average measure across all top m positions (Average Precision) for user u is defined as $(\sum_{k=1}^{m} \text{Precision@k} \times l_k)/(\#$ of clicks for ranked list of user $u)$ where l_k is a binary indicator whether the position k has been clicked or not and m is the total number of positions evaluated. Note that the Average Precision is evaluated per user. We can average it across all users, resulting in the Mean Average Precision (MAP) measure.

6.2.2 Dataset

Before discussing the details of our dataset, we introduce the concept of *impression*. Every time a user logs in LinkedIn's homepage, the system generates a list of candidate social updates from many sources, mainly based on the user's social connections. This list of updates can be small or large, depending on the user's social circle. If the number of updates in the list exceeds a certain threshold (e.g., $10 - 15$), these updates cannot be shown on a single screen, and the user will need to scroll down. An impression is a list of updates that a user has actually seen on the screen. Given historical data, we can "replay"

the users' activity while analyzing impressions. Note that social updates are not distinct: one specific update produced by a user or a company can be shown to many users.

Since our experimental setting is "simulation" (details will be discussed in Section 6.2.1), we discard all impressions without any clicks because these impressions do not change our experimental results (as measured by MAP). Note that there is a deeper argument on this decision. Remember that one issue associated with social streams is their sparsity. Indeed, only a small number of impressions attract users and a handful of clicks is performed on such impressions, compared to the large amount of impressions produced in total. Thus, it might be useless for any model to fit these non-interacted impressions. Focusing on impressions that actually matter reduces the training set significantly and produces better results, which we saw in our empirical studies. Thus, only impressions with at least one click remain in our dataset. In our experiments, we also filter out impressions with less than five items.

We report on two datasets of LinkedIn's social update stream. Both are subsamples of the actual social stream collected by LinkedIn's engineering team. The first dataset was taken from April, 2011 stream, while the second was taken from September, 2011 stream. The basic statistics on the two datasets are shown in Table 6.1 where "M" means million. The numbers are obfuscated due to commercial reason. The reason why we take two datasets that are not consequent in time is to demonstrate that the performance of different algorithms is indeed consistent over different time periods.

6.2.3 Linear Models

In this section, we will discuss several simple linear models to tackle the problem of SSR. Given a social update, a user can choose to respond to this update or not. For simplicity, we treat all kinds of responses as a "click" event and no response as a "non-click" event. Thus, we focus on binary responses in this work. We denote $\mathbf{y}$ as a vector of responses

Data Summary	April, 2011	September, 2011
Impressions	3M-4M	10M-20M
Updates	30M-40M	100M-200M
Clicked Updates	3M-4M	10M-20M
Non-clicked Updates	27M-36M	90M-180M
Distinct Updates	10M-20M	20M-30M
Recipients	1M-2M	4M-5M
Producers	4M-5M	6M-7M

Table 6.1: The basic statistics about the dataset.

to all social updates, across all impressions. This way, we concatenate updates from all impressions together and drop the notion of an impression. The ordering of elements in $\mathbf{y}$ does not matter as we only care about the correspondence: the response y_i corresponds to the i-th update in the entire dataset and f_i represents the estimation of y_i from the models described later. In addition, we define the following auxiliary functions: $r(i)$ is the recipient of update i, $s(i)$ is the sender of update i, $t(i)$ is the type of update i, and $c(i)$ is the sender type. Let $\mathcal{R}$ be the set of recipients, $\mathcal{S}$ be the set of senders, $\mathcal{T}$ be the set of types, $\mathcal{I}$ be the set of social updates.

Feature Model (FM): One straightforward model is linear estimation, which predicts the response by a linear combination of features. For a specific update i, we collect their corresponding features. Let $\boldsymbol{\phi}$ be a feature vector – we use subscript to indicate its corresponding type. For instance, $\boldsymbol{\phi}_{r(i)}$ represents the feature vector (e.g., profile) for the recipient user of update i. A simple prediction of y_i – a linear combination of user features and update features – can be defined as:

$$f_i^{(1)} = \boldsymbol{\beta}_{r(i)}^T \boldsymbol{\phi}_{r(i)} + \boldsymbol{\alpha}_{r(i)}^T \boldsymbol{\phi}_i \tag{6.1}$$

where $\boldsymbol{\beta}_u$ and $\boldsymbol{\alpha}_u$ are per-user coefficients to be learned from the training set. This model is essentially equivalent to the one where user features and update features are combined

into a single feature vector and a per-user coefficient to be learned. Note, an even more simpler model could be also considered where a universal coefficient is used, instead of per-user coefficients. However, this model would be too restricted and personalization cannot be applied.

Latent Bias Model (LBM): Here, we introduce a linear model to explain the clicks on items. We start with an assumption that whether a new item i will be clicked by a user depends on the average click rate: $f_i = \mu$ where μ is the average click rate across all items and all users. Certainly, this estimation is too coarse and inaccurate because, as we mentioned before, the majority of items are not clicked. We can extend this base estimation by incorporating a wide range of biases: 1) type (category) bias, 2) item bias, 3) recipient bias, 4) sender type bias and 5) sender bias. Adding these biases is very intuitive. Certain types of updates (e.g., notifications about changes in user profiles, including changes in job titles etc.) receive more attentions than others. Users tend to respond (e.g., click "Like" button or make comments) more often on these types of updates than on others. In addition, some individual items are more popular than others as their content might be more interesting (e.g., breaking news, unexpected stories). From the perspective of senders and recipients, biases are also significant. For instance, some updates are coming from companies that inform their followers about their new products and services – those updates are far more popular than status updates from individual users. Moreover, certain users are more engaged than others which introduces user biases. Therefore, all these biases (i.e., prior knowledge) can capture a wide range of effects of interactions. Let b denote biases and the subscript to indicate the type of biases. Also, the subscript can be an index for a feature. Therefore, we can have the following linear estimation:

$$f_i^{(2)} = \mu + b_i + b_{t(i)} + b_{r(i)} + b_{c(i)} + b_{s(i)} \tag{6.2}$$

Note that these biases are generally unknown. We treat them as latent variables to be learned from the dataset. Comparing with LR, which depends on feature vectors some of which might be difficult to calculate and update (e.g., graph-based features, content based features), this model is appealing since no extra information is needed for learning, besides requiring indicators.

Combining FM and LBM: It is straightforward to consider combining FM and LBM together. Thus, the combined model will enjoy the freedom that different parts of the model will explain a variety of user behaviors. The combined model is simply:

$$f_i^{(3)} = f_i^{(1)} + f_i^{(2)} \tag{6.3}$$

Note, this is essentially a linear feature model with biases decomposed into many aspects.

Incorporating Temporal Effects: Social streams are temporally sensitive in nature. Users focus on fresh information and interact with them while they often do not bother to look at old updates. We model the temporal effects of updates by a simple feature: $t_{\text{recency}} = t_{\text{imp}} - t_{\text{upt}}$ where t_{imp} is the numerical time when a user sees a particular impression and t_{upt} is the numerical time when an update is produced. These feature values would be different for different updates. Even for the same update, depending on when recipients access their update streams, the feature value can vary greatly. Incorporating this feature into our estimation can be as follows:

$$f_i^4 = f_i^{(*)} + \zeta \times t_{\text{recency}} \tag{6.4}$$

where $f_i^{(*)}$ can be any estimator defined in Equations (6.1, 6.2 and 6.3) and ζ is a free parameter, indicating the importance of recency of updates. Note that ζ can be learned from the training set and can also be treated as another personalized parameter. However,

we fix it across all users and manually tune this parameter mainly because of two reasons. Firstly, since not all users interact with social streams regularly and new users are coming all the time, we need to provide reasonably relevant social updates for these users. Under these circumstances, explicit features might not be available (e.g., new users) and biases are not learned reliably from the training set (e.g., not enough interactions before). Therefore, a safe choice is to rank items chronologically. In addition to that, users who are heavily engaged with their social streams are familiar with existing ranking schemes which are primarily based on recency. We do not want to suddenly change their expectations on their future impressions. Hence, we set ζ such that the temporal effect can always be an important factor and indeed the learned coefficients and biases will dominate the estimation only when it is necessary.

Since responses are binary, we impose a logistic loss on predictions and true values, yielding learning procedures of the logistic regression flavor. All these linear models can be learned effectively by minimizing the following objective function for each update i:

$$l_1(y_i, f_i^{(*)}) = \log\Big[1 + \exp(-y_i f_i^*)\Big] \tag{6.5}$$

where $y_i \in \{\pm 1\}$ is the ground-truth response and f_i^* is the estimated response from the models defined above. In common practice, in order to avoid overfitting the training set, we also use L_2 regularizer to shrink all parameters towards zero. Taking Equation (6.4) as an example, the final objective function is:

$$\begin{aligned} L_1 = \sum_i l_1(y_i, f_i^4) + \lambda_1 \Big(\sum_i ||b_i||^2 + \sum_{t(i)} ||b_{t(i)}||^2 + \sum_{c(i)} ||b_{c(i)}||^2 \\ + \sum_{r(i)} ||b_{r(i)}||^2 + \sum_{s(i)} ||b_{s(i)}||^2 \Big) + \lambda_2 \sum_u \Big(||\boldsymbol{\beta}_u||_F^2 + ||\boldsymbol{\alpha}_u||_F^2 \Big) \end{aligned}$$

where λ_1 and λ_2 are two regularization parameters to be manually tuned. Many methods are available to optimize the objective function above. Here, we adopt the Stochastic

Gradient Descent (SGD) method, a widely used learning method for large-scale data, to learn parameters. SGD requires gradients, which can be effectively calculated as follows:

$$\frac{\partial L_1}{\partial b_*} = -\sum_i \Big[1 - \sigma(y_i f_i^4)\Big] y_i + 2\lambda_1 \sum_* b_*$$
$$\frac{\partial L_1}{\partial \boldsymbol{\beta}_{r(i),k}} = -\sum_i \Big[1 - \sigma(y_i f_i^4)\Big] y_i \boldsymbol{\phi}_{r(i),k} + 2\lambda_2 \boldsymbol{\beta}_{r(i),k}$$
$$\frac{\partial L_1}{\partial \boldsymbol{\alpha}_{i,k}} = -\sum_i \Big[1 - \sigma(y_i f_i^4)\Big] y_i \boldsymbol{\phi}_{i,k} + 2\lambda_2 \boldsymbol{\alpha}_{i,k}$$

where b_* represents any bias terms, $\boldsymbol{\beta}_{r(i),k}$ and $\boldsymbol{\alpha}_{i,k}$ represent k-th element of coefficient for user $r(i)$ and update i respectively. Note that $\sigma(x) = \frac{1}{1+\exp(-x)}$.

6.2.4 Latent Factor Models

Although linear models are efficient, they are usually oversimplified and cannot capture interactions between different effects. Latent Factor Models (LFM) are widely used in recommender systems (e.g., [121, 211, 213, 214]) and have proven effective in many scenarios (e.g., [121]). Specifically, LFM can model the interactions between different types of entities such as user-user and user-item, discovering their latent relationships. In this section, we discuss two types of LFM: matrix factorization and tensor factorization, and see how they can be applied to the task of SSR.

Matrix Factorization: In traditional CF, matrix factorization techniques are used to exploit user-item interactions. A straightforward idea would be to directly apply matrix factorization methods to user-update matrix. However, this idea is not practical. As discussed above, social streams have new updates arriving all the time, and existing updates are only consumed by a small number of users. Thus, cold-start problems are much more severe here, compared to traditional CF where the user base and the item base are relatively stable. Here, we impose a latent factor $\boldsymbol{\eta}_u \in \mathbb{R}^k$ for each recipient and

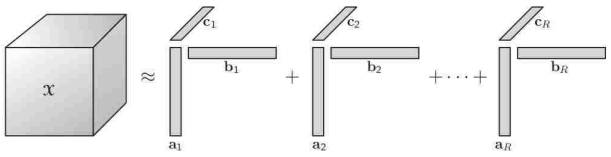

Figure 6.2: CANDECOMP/PARAFAC decomposition of a tensor, a three-way array.

producer and factorize the recipient-producer matrix to predict the actions on updates. We describe the model in a probabilistic way:

$$\boldsymbol{\eta}_u \sim P(\boldsymbol{\eta}_u \,|\, \mathbf{0}, \sigma^2 \mathbf{I}) \;\; u \in \{\mathcal{R}, \mathcal{S}\}$$
$$y_i \sim P(y_i \,|\, \boldsymbol{\eta}_{r(i)}, \boldsymbol{\eta}_{s(i)}, b_*, \mu)$$

where b_* is any biases introduced in Section 6.2.3, $\mathcal{R}$ and $\mathcal{S}$ are the set of recipients and producers respectively. For $P(\boldsymbol{\eta}_u \,|\, \mathbf{0}, \sigma^2 \mathbf{I})$, it is usually assumed to be Gaussian or Laplace, corresponding to L_2 or L_1 regularization on latent factors respectively. Here, we use a multivariate Gaussian assumption. For $P(y_i \,|\, \boldsymbol{\eta}_{r(i)}, \boldsymbol{\eta}_{s(i)}, b_*, \mu)$, we assume:

$$y_i \sim P(y_i \,|\, f_i)$$
$$f_i = \mu + b_i + b_{t(i)} + b_{r(i)} + b_{c(i)} + b_{s(i)} + \boldsymbol{\eta}_{r(i)}^T \boldsymbol{\eta}_{s(i)} + \epsilon$$

where ϵ allows a Gaussian distribution. Thus, the final generative process also follows a Gaussian distribution. This formalism is similar to the one introduced in [121]. The model described here is very intuitive. Whether a user u_1 is going to click on an update from a user/company u_2 depends on u_1's and u_2's affinity.

Tensor Factorization: As social streams have different entities like recipients, producers and categories, it would be natural to directly model the multi-way data interaction, rather than concentrating on two-way relationships. It has been shown that high-order

relational modeling can improve the performance of CF systems in many scenarios, for instance in social tag recommendations [166, 168].Here, we focus on one particular three-way relationship: recipient-producer-category of the update. We associate latent factors $\boldsymbol{\eta}_x \in \mathbb{R}^k$ for these three types of entities. Similar to the matrix case, we define the following generative procedures:

$$\boldsymbol{\eta}_x \sim P(\boldsymbol{\eta}_x \,|\, \mathbf{0}, \sigma^2 \mathbf{I}) \;\; x \in \{\mathcal{R}, \mathcal{S}, \mathcal{T}\}$$
$$y_i \sim P(y_i \,|\, f_i)$$

where f_i is defined as:

$$f_i = \mu + b_i + b_{t(i)} + b_{r(i)} + b_{c(i)} + b_{s(i)} + \sum_k \boldsymbol{\eta}_{r(i),k} \boldsymbol{\eta}_{s(i),k} \boldsymbol{\eta}_{t(i),k} + \epsilon$$

where $\boldsymbol{\eta}_{*,k}$ represents the k-th element in the vector and ϵ again follows a Gaussian distribution. This particular form of tensor decomposition is known as CANDECOMP/PARAFAC (CP) decomposition [120], depicted in Figure 6.2 where the dimensionality of three latent factors is the same. There are two important properties about CP decomposition. Firstly, it is a direct analogue to factorization methods in two-way array (matrix) data where latent factors share the same latent space. Secondly, CP decomposition has a nice yet surprising property that it has a unique solution of decomposition where matrix factorization does not enjoy this result [120]. This property indeed provides a theoretical guarantee to the decomposition and may be a reason for better performance.

We are aware of other forms of tensor factorization as well. For instance, Tucker decomposition [120], where the dimensionality of different latent factors varies, is widely used in many applications and applied to social media as well (e.g., [166, 211]). However, we do not choose the Tucker decomposition for our settings because not only it requires to pre-specify the dimensionality of all factors separately, but also does not guarantee

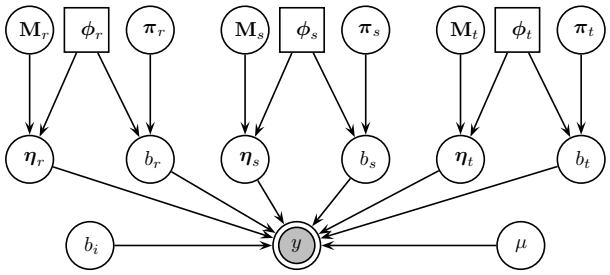

Figure 6.3: A graphical representation of regression-based tensor factor model.

uniqueness of the decomposition result.

Incorporating Features: Both matrix factorization and tensor factorization discussed above do not directly incorporate explicit features. Here, we introduce features into the model by employing two levels of regression models. The basic idea is that latent features will depend on explicit features and final responses are derived from latent features. The first level regression models are defined as:

$$\boldsymbol{\eta}_{x(*)} = \mathbf{M}_x \boldsymbol{\phi}_{x(*)} + \epsilon_x \qquad\qquad x \in \{\mathcal{R}, \mathcal{S}, \mathcal{T}\}$$

where $\boldsymbol{\phi}_{x(*)}$ represents a feature vector for entity x and $\mathbf{M}_x$ is a transformation matrix to be learned. If ϵ_* follows a zero-mean k-dimensional Gaussian distribution, latent factors $\boldsymbol{\eta}_*$ indeed follow multivariate Gaussian distribution with the mean of a transformation of explicit feature vectors. This way, explicit feature space is mapped to latent feature space.

In addition to binding latent factors to explicit features, other biases may also have the same prior distributions:

$$b_{x(*)} = \boldsymbol{\pi}_x^T \boldsymbol{\phi}_{x(*)} + \epsilon_{b_x}$$

where $\boldsymbol{\pi}_x$ is a regression coefficient for entity x. If the error term ϵ_{b_x} follows a Gaussian

distribution, biases $b_{x(*)}$ will also follow a Gaussian distribution centered at a transformation from explicit features. Note that this two-level regression scheme can be applied to matrix factorization as well as tensor factorization. The idea to use regression priors for matrix factorization has been explored by [1, 214] but not yet discussed on multi-way data relations like tensors. The final graphical representation of the model is shown in Figure 6.3 where square nodes represent features and circled nodes represents unknown variables. The response y in the middle is observed in the training set but to be predicted in the test set.

In addition to the method described here to incorporate features, we are aware of other possibilities, such as [165] where latent factors and explicit features are treated as same set of features.

Like linear models from Section 6.2.3, LFM (with features) can also be learned through a Maximum A Posterior (MAP) estimation. Taking Tensor Factorization with Features as an example, the problem of minimizing the negative log posterior of the model boils down to the following objective:

$$
\begin{aligned}
L_2 = & \sum_i L_1(y_i, f_i) + \sum_{x \in \{\mathcal{R}, \mathcal{S}, \mathcal{T}\}} \lambda_x \sum_{x(i)} ||\boldsymbol{\eta}_{x(i)} - \mathbf{M}_x \boldsymbol{\phi}_{x(i)}||_F^2 \\
& + \sum_{x \in \{\mathcal{I}, \mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{C}\}} \lambda_{b_x} \sum_{x(i)} ||b_{x(i)} - \boldsymbol{\pi}_x^T \boldsymbol{\phi}_{x(i)}||^2 \\
& + \sum_{x \in \{\mathcal{I}, \mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{C}\}} \left(\lambda_{\pi_x} ||\boldsymbol{\pi}_x||_F^2 + \lambda_{M_x} ||\mathbf{M}||_F^2 \right)
\end{aligned}
$$

where all constant terms are ignored and all λ terms are manually tuned regularization parameters. For both matrix factorization and CP decomposition, a number of techniques are available to solve the objective function. For instance, the alternating least squares (ALS) method is the "workhorse" [120] for both matrix and tensor factorization. However, here, we still adopt a SGD method, which can scale to the dataset with which we are

working. In order to use SGD, the gradients of latent factors can be derived. Firstly, we focus on latent factors:

$$\begin{aligned}
\frac{\partial L_2}{\partial \boldsymbol{\eta}_{r(i),k}} &= -\sum_i \Big[1-\sigma(y_i f_i)\Big] y_i \boldsymbol{\eta}_{s(i),k} \boldsymbol{\eta}_{t(i),k} \\
&\quad + 2\lambda_r \Big(\boldsymbol{\eta}_{r(i),k} - \mathbf{M}_{r[k]} \boldsymbol{\phi}_{r(i)}\Big) \\
\frac{\partial L_2}{\partial \boldsymbol{\eta}_{t(i),k}} &= -\sum_i \Big[1-\sigma(y_i f_i)\Big] y_i \boldsymbol{\eta}_{r(i),k} \boldsymbol{\eta}_{s(i),k} \\
&\quad + 2\lambda_t \Big(\boldsymbol{\eta}_{t(i),k} - \mathbf{M}_{t[k]} \boldsymbol{\phi}_{t(i)}\Big) \\
\frac{\partial L_2}{\partial \boldsymbol{\eta}_{s(i),k}} &= -\sum_i \Big[1-\sigma(y_i f_i)\Big] y_i \boldsymbol{\eta}_{r(i),k} \boldsymbol{\eta}_{t(i),k} \\
&\quad + 2\lambda_s \Big(\boldsymbol{\eta}_{s(i),k} - \mathbf{M}_{s[k]} \boldsymbol{\phi}_{s(i)}\Big)
\end{aligned}$$

where $\boldsymbol{\eta}_{*,k}$ is the k-th element of the vector and $\mathbf{M}_{*[k]}$ is the k-th row of the matrix. For all biases, gradients $\frac{\partial L_2}{\partial b_{*(i)}}$ are as follows:

$$-\sum_{i \in b_{*(i)}} \Big[1-\sigma(y_i f_i)\Big] y_i + 2\lambda_{b_*} \Big(b_{*(i)} - \boldsymbol{\pi}_*^T \boldsymbol{\phi}_{*(i)}\Big)$$

where $*$ means a particular type of bias and $i \in b_{*(i)}$ represents the updates those bias type match the type of interests. The gradients $\frac{\partial L_2}{\partial M_{x[k,m]}}$ for matrix M_x can be derived as:

$$2\lambda_x \sum_{x(i)} \Big(\boldsymbol{\eta}_{x(i),k} - \mathbf{M}_{x[k]} \boldsymbol{\phi}_{x(i)}\Big)\Big(-\boldsymbol{\phi}_{x(i),k}\Big) + 2\lambda_{M_x} \mathbf{M}_{x[k,m]}$$

where $\mathbf{M}_{x[k,m]}$ is the (k,m)-th element in the matrix $\mathbf{M}_x$. And finally, the gradients for regression coefficients $\frac{\partial L_2}{\partial \boldsymbol{\pi}_{x,k}}$ can be computed as:

$$2\lambda_{b_x} \sum_{x(i)} \Big(b_{x(i)} - \boldsymbol{\pi}_x^T \boldsymbol{\phi}_{x(i)}\Big)\Big(-\boldsymbol{\phi}_{x(i),k}\Big) + 2\lambda_{\pi_x} \boldsymbol{\pi}_{x,k}$$

6.2.5 Pairwise Learning

So far, we have demonstrated two different types of models: linear models and latent factor models. Both of them minimize certain errors in the learning process. As discussed in Section 6.2.1, a ranking-based evaluation metric, MAP is used in our experiments. Thus, it is more reasonable to directly optimize this ranking metric. However, it is difficult to optimize ranking measures directly [221, 42, 162] due to their discrete nature. Although some techniques (e.g., [42, 162]) have been developed to derive smoothed surrogate functions to approximate these ranking measures, including MAP, they are usually complicated and expensive to apply to large scale scenarios. Here, we use a much simpler approach: derive pairwise preferences from users' impressions and learn a pairwise ranking function.

First, let $\mathcal{O}_i$ be the set of updates in the impression i, $\mathcal{O}_{i,+}$ be the set of updates clicked by the user and $\mathcal{O}_{i,-}$ be the set of updates not clicked by the user. Remember that we eliminate impressions without any clicks (see 6.2.2). Therefore, we guarantee that the method described here can be applied to all impressions in our dataset. For any pair of updates (m, n) where $m \in \mathcal{O}_{i,+}$ and $n \in \mathcal{O}_{i,-}$, we can always construct a preference label $l_{m,n} = 1$, meaning that update m is favored over update n in impression i. Under this setting, we have the new objective function for impression i:

$$l_2(i, \mathbf{f}) = \frac{1}{|\mathcal{O}_{i,+}||\mathcal{O}_{i,-}|} \sum_{m \in \mathcal{O}_{i,+}} \sum_{n \in \mathcal{O}_{i,-}} \sigma\Big(f_m - f_n\Big) \tag{6.6}$$

where σ is a logit function. This new objective function is no longer to fit a single observed label (click or not) but to optimize a pairwise preference induced from impressions. Similar ideas are also explored in [167, 213, 122]. With this new objective function, we can replace the original loss function defined in Equation (6.5) in both linear model learning and factor model learning. Gradients are omitted due to space limits.

6.2.6 Summary & Discussion

We discussed several issues related to our proposed methods in this sub-section: 1) parameter tuning, 2) scalability and 3) feature treatment. For parameter tuning, while it is not a significant problem for Linear Models, as they can be trained efficiently, it might be prohibitively expensive to tune a Latent Factor Model. In this work, we do not heavily tune parameters and only wish to see whether these proposed approaches work in principle. One way to deploy a "parameter-free" model might be to consider a Bayesian treatment of Latent Factor Models, like [175, 211]. However, the sheer amount of data and its continuous nature prevent us to explore Bayesian treatment in this work and leave it to the future work. In terms of scalability, we conduct experiments on a single machine in this work but we do notice that SGD can be paralleled [232]. Thus, we can scale our model to even larger datasets. The way we integrate explicit features and latent factors is through regression models. However, this is not the only way to deal with this kind of problem. For instance, matrix co-factorization (see, e.g., [181]) and tensor co-factorization can be another paradigm of combining explicit features and hidden features.

6.3 Experimental Results

In this section, we demonstrate the effectiveness of our model, through a comprehensive comparison with non-trivial baselines. The dataset used in our experiments is described in Section 6.2.2. Before we go into the details of the experimental results, we first discuss our experimental setting in Section 6.3.1 and then all developed models in Section 6.3.2.

6.3.1 Experimental Setting

Two standard settings are available to evaluate the effectiveness of systems for SSR. One is to test each model against an existing system in a online setting where both systems run in

parallel for a similar audience in a reasonable period of time. After this, the effectiveness of both systems can be calculated using certain measurements, like error rate or ranking metrics. This is a classic A/B testing scenario. The advantage of A/B testing is obvious: it provides a real comparison between models. However, it might be time-consuming and even impossible to compare many models in a batch. In addition, some models require tuning parameters, which may risk the business of the service a company offers. Thus, we do not use A/B testing in this paper and leave it to the future work.

In this paper, we simulate real settings of SSR, conducting off-line experiments. More specifically, we gather historical data from LinkedIn user logs, which capturing all impressions users have consumed. Since we know which updates are clicked in each impression, it is easy to replay all these impressions and reorder the updates. Thus, we can produce a "new" impression for users in the dataset. The drawback of this approach to experiments is that we cannot show "new" ordering of impressions that are not clicked by users at all because whether users would have clicked them or not is impossible to test. This is another reason why we drop all impressions without any clicks (Section 6.2.2).

The dataset for one month is divided into weeks. We train our models on one week and test them on the following week. This setting results in more than 70% items being new in test data each week, which is an evidence to the fact that SSR is different from RecSys and IR.

6.3.2 Models & Features

We compare several models in our experiments. All models used in the following experiments are shown in Table 6.2. The baseline is a proprietary system currently deployed in the product of LinkedIn homepage. `FM`, `LBM` and their combination (`FBM`) are examples of simple linear models while `MF`, `TF` with their feature enhanced extensions (`MF2` and `TF2`) are examples of latent factor models. For all models, a point-wise loss function (Equation

Models	Comments
Baseline (BL)	LinkedIn
Feature Model (FM)	Section 6.2.3
Latent Bias Model (LBM)	Section 6.2.3
Feature Bias Model (FBM)	Section 6.2.3
Matrix Factorization (MF)	Section 6.2.4
Tensor Factorization (TF)	Section 6.2.4
Matrix Factorization with Features (MF2)	Section 6.2.4
Tensor Factorization with Features (TF2)	Section 6.2.4

Table 6.2: All models used in our experiments.

Features	Comments
Seniority	the seniority level of a user
Visiting	how frequently a user visits LinkedIn
PageRank	discretized PageRank scores
Connectedness	how well a user is connected to others
Social strength	how tight a user's connections is
Professional	how professional an update's language is
Recency	the freshness of an update (see Section 6.2.3)

Table 6.3: All features used in our experiments.

(6.5)) and a pairwise loss function (6.6) are both tested. Without stating it explicitly, all models include the temporal effect feature discussed in Section 6.2.3 while the parameter ζ, the balance between recency and relevance, is manually tuned. All regularization parameters are simply set to 1. We understand that this may not be an optimal choice. For tuning a reasonable learning rate in SGD and ζ, we use the first day in the test week as a "validation" day and choose the parameter setting that can provide the optimal performance on the day. We fix the parameters for the remaining days in the test week.

Some of linear models and latent factor models require explicit features. In this paper, we include several important features to enhance our models. Note that we are aware of many other possible features. However, it is not our goal to study the effectiveness

Training/Testing	BL	FM	LBM	FBM
4_01(Tr.)/4_08(Te.)	0.5278	0.5317	**0.5943**	0.5520
4_08(Tr.)/4_15(Te.)	0.5435	0.5509	**0.6040**	0.5574
4_15(Tr.)/4_22(Te.)	0.5218	0.5246	**0.5823**	0.5235
9_01(Tr.)/9_10(Te.)	0.4829	0.4911	**0.5457**	0.4984
9_10(Tr.)/9_18(Te.)	0.4779	0.4798	**0.5432**	0.4915
9_18(Tr.)/9_25(Te.)	0.4768	0.4803	**0.5329**	0.4886

Table 6.4: The comparison between linear models.

of a particular feature in this work. All features are shown in Table 6.3. "Seniority" measures the seniority level of a user's job title. "Visiting" measures how well engaged a user is (our assumption is that a frequent visitor is likely to interact with his/her social stream). "PageRank" (details in [33]) and "Connectedness" measure how a user connects with other users. Presumably, a highly respected and well connected user can attract others to interact with their update streams. "Social strength" is a proprietary product used in LinkedIn, measuring the connection closeness between two users. "Professional" measures how likely an update is similar in its language to professional profiles of LinkedIn users (i.e. how professional an update is). The assumption is that users may favor professional updates over non-professional updates on LinkedIn because it is a professional social network.

6.3.3 Results on Linear Models

In this sub-section, we focus on the comparison between the baseline and all linear models.

In this Subsection, we focus on the comparison between the baseline and all linear models. The results are shown in Table 6.4 where the best performance is shown in bold. The first column indicates how models are trained and tested. For instance, the first number "4_01" means the models are trained on the week of April 1st and tested on the week of April 8th (8-th is the date for validation of parameter tuning) where "Tr." and "Te." are

Type Description	**Bias** b_t
Job Seeker Product Update	0.5765
Joining Sub-Group	0.5407
Company News	0.4592
Joining Group	0.2625
Profile Picture Update	0.2516
Initiating Direct Ads Campaign	0.2253
Profile Update	0.1394

Table 6.5: Example of highly ranked types of updates

shorthand for "Training" and "Testing", respectively. We conduct experiments on two separate months to avoid some seasonal fluctuations on the data. The numbers shown on the right part of the table are MAP scores.

Our first observation is that the baseline of MAP in September is lower than its in April, implying that updates in the lower positions in the lists get clicked more often over time. One possible explanation is that users become familiar with their social streams and start to find interesting updates manually, looking at more items down the list. The second observation is that all linear models, including `FM`, `LBM` and `FBM`, perform better than the baseline, consistently on two-month datasets. However, for `FM`, which only depends on explicit features, the performance is very close to the baseline. This is reasonable because only a handful of features are used in our experiments and we do expect that these features are not likely discriminative. On the other hand, `LBM`, a model only depending on implicit feedback, has consistently $5\% - 6\%$ absolute improvements on MAP over the baseline. This confirms that it is vital to exploit different aspects of users' feedbacks and capture them through bias modeling (e.g., [121, 119]). Indeed, `FBM` gives the most improvements over the baseline in all our experiments. The idea is simple and easy to implement. For the combination of a pure feature-based model `FM` and a pure implicit-feedback-based model `LBM`, `FBM` does perform as someone might expect. It is significantly worse than `LBM` and

Training/Testing	MF	TF	MF2	TF2
4_01(Tr.)/4_08(Te.)	0.5955	0.6258	0.5951	**0.6336**
4_08(Tr.)/4_15(Te.)	0.6079	0.6228	0.6088	**0.6535**
4_15(Tr.)/4_22(Te.)	0.5962	0.6014	0.5991	**0.6312**
9_01(Tr.)/9_10(Te.)	0.5511	0.5766	0.5523	**0.6003**
9_10(Tr.)/9_18(Te.)	0.5412	0.5833	0.5449	**0.6109**
9_18(Tr.)/9_25(Te.)	0.5359	0.5799	0.5362	**0.5992**

Table 6.6: The comparison between latent factor models.

almost identical to FM, which might indicate that simply integrating explicit features with biases may not be a good choice and more sophisticated approaches are needed.

LBM can also reveal some interesting patterns from the dataset, which might not be easily identified by other methods. For instance, we can figure out the effective popularity of different types of updates by looking at the values of b_t. The positivity or negativity of these values indicate whether a particular type of update correlates with clicks or non-clicks, while the magnitude of these values means how strongly this correlation might be. We show some samples of top ranked types in Table 6.5, which are positively correlated with clicks. From the table, we see that job related updates, company-related updates are comparatively attractive. In addition, users pay attention to profile changes of their connections and new connections established by their friends. Note that the value shown in the table is "automatically" normalized in the sense that SGD only updates corresponding parameters when the algorithm meets such type of updates. Also, the ratio of positive examples of a particular type will drive the parameter to strong positive numbers. Thus, no post-processing steps are required. This is an example of how our model can be used in simple data analysis tasks.

6.3.4 Results on Latent Factor Models

As we discussed earlier, latent factor models are widely used and have been proven to be superior to linear models. We conduct the same experiments as linear models and show their results in Table 6.6 where the best performance is shown in bold. Here, we compare between pure factorization-based models (`MF` and `TF`) and feature-enhanced factor models (`MF2` and `TF2`). Note that all these models are built upon `LBM` and therefore the performance should at least match `LBM`. The second column of the table shows the results from `MF`, which is essentially to factorize the recipient-sender matrix and uncover latent structures. Unfortunately, the gain of performance of `MF` over `LBM` is marginal and even not observable. On the other hand, `TF` offers significant improvements and leads another $3\% - 4\%$ absolute boost for MAP on average. As we discussed before, social stream data is much more complex than traditional recommender system data (in various CF scenarios). Users may interact with certain updates because their senders are famous people or because the type of updates (e.g., news or twitter updates) is of a particular interest. Thus, tensor factorization can model multi-way relationships better than matrix factorization models them via a decomposition to multiple two-way relationships. Indeed, bias terms of recipients and senders in `LBM` might capture the basic relationship between them and a matrix factorization does not provide any additional benefits. Furthermore, we have already discussed why we do not employ the user-item matrix in our setting: new items are much more common in social streams, compared to other recommender system setups. Thus, it is very interesting to see that well-established matrix-based factor models do not work equally well on social streams as they do in traditional CF scenarios. We believe that a more thorough investigation on this issue is desired.

For latent factor models with features, it is noticeable that `MF2` failed to outperform `LBM` again while `TF2` has gained additional $2\% - 3\%$ absolute improvement over `TF` consistently. This is a yet another signal that matrix factorization does not work well in SSR. For `TF2`,

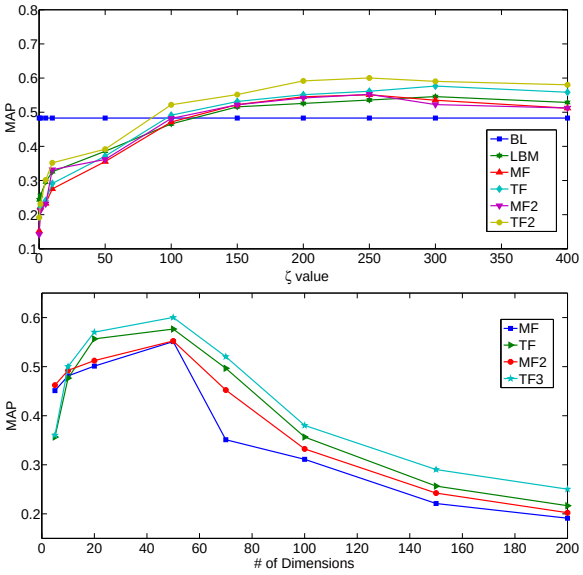

Figure 6.4: Parameter Sensitivity Analysis.

the increase of MAP can be explained by the two-level regression structure, that explicit features will help create latent features when new items or new users come into the system, effectively mitigating the cold-start problem. The performance of `TF2` also validates that regression-based latent factor models are an effective approach to integrate explicit features with latent features. We also study the sensitivity of parameters, especially the temporal balance weight ζ and the number of dimensions in latent factor models. Taking the first week of September as an example, the effects of both parameters are shown in Figure 6.4 where the effect of ζ is shown on the top and the effect of # of dimensions in latent factor models is shown on the bottom. The first observation is that both parameters are vital to the final performance and they are very sensitive if they are out of certain ranges. For ζ, the optimal performance is achieved when it is around $250 - 300$ while for dimensionality, the results suggest that a reasonable number $(20 - 50)$ is the key to success.

Training/Testing	LBM	MF	MF2	TF	TF2
4_01(Tr.)/4_08(Te.)	0.6169	0.6033	0.6151	0.6358	**0.6532**
4_08(Tr.)/4_15(Te.)	0.6188	0.6168	0.6188	0.6528	**0.6641**
4_15(Tr.)/4_22(Te.)	0.5897	0.6104	0.6191	0.6014	**0.6402**
9_01(Tr.)/9_10(Te.)	0.5644	0.5716	0.5723	0.5966	**0.6207**
9_10(Tr.)/9_18(Te.)	0.5593	0.5621	0.5607	0.5999	**0.6183**

Table 6.7: The effects of pair-wise learning.

6.3.5 Results on Pairwise Learning

The results demonstrated so far focus on point-wise learning procedure. In other words, the objective function imposed by models is still error-based loss function. Here, we focus on how to improve performance by switching the objective function to pairwise preferences learning. More specifically, we conduct similar experiments as previous ones and report results on LBM, MF, TF MF2 and TF2, shown in Table 6.7 where the best performance is shown in bold. Other models are omitted due to their poor performance. First of all, we notice that almost all models can benefit from pairwise learning, even for the methods which did not show significant gains in previous experiments, such as MF and MF2. However, on another perspective, the overall improvement of a pairwise learning is not huge, usually yielding $1.5\% - 2\%$ improvement on MAP. One possibility is that the pairwise learning here is still naïve. More sophisticated session enabled learning procedures (e.g., [213]) are to be investigated in the future.

6.4 Summary

In this chapter, we investigate the problem of ranking social updates from a unique perspective of LinkedIn, the largest professional social network in the world. More specifically, we address the task as an intersection of learning to rank, collaborative filtering and click-through modeling, leveraging ideas from information retrieval and recommender systems.

We propose a novel probabilistic latent factor model with regressions on explicit features, comparing a number of non-trivial baselines and gaining an approximately 10% improvement on MAP over the baseline. In addition to superior performance demonstrated in the paper, we shed some light on social updates on LinkedIn and how users interact with them, which might be applicable for social streams in general. For future work, it is interesting to see whether it is possible to develop efficient Bayesian treatment of latent models. In addition, other models might be explored as we demonstrate that state-of-the-art CF models do not provide comparable success in SSR. We also wish to extend our work by considering the diversity of information users wish to consume.

6.5 Bibliographic Notes

In this section, we briefly overview three research directions related to social stream ranking: (a) learning to rank, (b) recommender systems, and (c) clickthrough models. Some of the approaches to tackle these problems are relevant to SSR and can be adapted. Along with their similarities to SSR, we also reveal their significant differences from SSR, and discuss the uniqueness of SSR as a new research field.

Learning to Rank (LtoR): In IR, a generic task is to construct a ranked list of documents relevant to a query issued by a user. Although ranking is a fundamental problem in IR and has been studied for decades, it still remains challenging. Instead of proposing carefully designed ranking models based on heuristics or traditional probabilistic principles, a recent trend is to apply machine learning techniques to learn ranking functions automatically, i.e., LtoR [137]. In the standard LtoR setting, a typical training set consists of queries with their associated documents represented by feature vectors as well as corresponding relevance judgements. A machine learning algorithm is employed to learn

the ranking model, which can predict the ground truth label in the training set as accurately as possible – in terms of a loss function. In the test phase, when a new query comes in, the learned model is applied to sort the documents according to their relevance to the query, and return the corresponding ranked list to the user as the response to the query. Depending on different hypotheses, input spaces, output spaces and loss functions, approaches to LtoR can be loosely grouped into three categories: point-wise, pairwise, and list-wise.

Although the goal of LtoR is to provide a ranked list of documents (items) for users – a similar aim as of SSR – it is different from SSR in three aspects. First, social stream systems usually do not have explicit queries and users do not have to specify any explicit input in order to obtain relevant output from the systems. Second, each user's social stream is highly dependent on their social context. Therefore, compared to IR, social streams are intrinsically personalized. The second difference leads to the third fundamental distinction between SSR and LtoR: relevance judgements are difficult to obtain in SSR and the notion of relevance can be hard to define. Nevertheless, some strategies and insights developed in LtoR can be borrowed for SSR.

Recommender Systems (RecSys): As we will see, RecSys plays a key role in many online services, improving user experience and engagement. In the simplest form, RecSys aim to present a user with a list of items in the hope that these items would match the user's interests to some extent. Content-based methods and neighborhood methods are widely used in RecSys. Content-based methods convert the problem of recommendation into an IR problem by constructing user profiles and item profiles. A user profile serves as a query to an index of item profiles. Similarity measures (e.g., cosine similarity, Jaccard coefficient) are utilized to match users and items. In contrast, neighborhood methods usually explore the notion of *topical locality*, assuming that the interaction between users and items can be predicted solely upon observations of "neighboring" users or items. That is, a user's

interest in an item is approximated by the average of neighboring observations. Although content-based methods and neighborhood methods are popular due to their simplicity, they cannot exploit hidden interactions between users and items. Recently, another class of methods called *latent factor models* has gained increasing attention. These methods are highly accurate and can easily incorporate various biases. However, compared to content-based methods and neighborhood methods, latent factor models are more vulnerable to the appearance of new items and new users, i.e., to the cold-start problem. Therefore, these three approaches are complementary to each other in practice (e.g., [121]). In a general sense, SSR may be considered as RecSys, however, SSR usually does not have item ratings, and user feedback to SSR is often implicit.

Click-through Model (CM): both IR (see, e.g., [109, 55]) and RecSys [167, 213] researchers have noticed that users' feedback is vital for learning a high-quality model. In order to derive users' preferences and model users' clickthrough data, a variety of CMs have been proposed. The most common approach to clickthrough modeling is to construct a generative model aiming to explain the training data (see, e.g., [225, 99]). Other models that derive users' preferences have been proposed as well (see, e.g., [167, 34, 213, 122]). While generative models are specifically designed to understand clickthrough data, it is difficult to incorporate them into current IR or RecSys frameworks, partially due to the fact that generative models are hard to adapt to optimizing a non-probabilistic objective. Indeed, it is easier to first obtain user preferences from clickthrough data analysis and then adapt existing IR/RecSys tools to using these preferences (e.g., [167, 34, 213]). Our paper is inspired from this idea.

In addition to these three directions, some efforts have been made to directly tackle the problem of SSR. For instance, Chen et al. [46] discussed the problem of recommending content on Twitter by considering many dimensions, including content sources, topic interests of users and social voting. However, their study focused on empirical validations

of several features (signals) and the dataset used is significantly smaller than ours. Their later work [45] goes beyond single Tweet recommendation to conversation recommendation. Duan et al. [63] noticed that the ranking problem of Twitter can be treated as an application of learning to rank. Their dataset is also small and relationships between recipients and senders are not explored. As we have discussed, SSR is not simply a LtoR problem. Choudhury et al. [57] argued that SSR should consider the problem of "diversity" and they tested their greedy algorithm on 67 employees from a large technology corporation. Our work differs from all this related work in three significant ways: 1) we test our proposed method on a large-scale, real-world dataset; 2) we propose a principled way to address SSR in the context of LtoR, CF and CM; and 3) we conduct comprehensive evaluation of our model against several models that underlie state-of-the-art recommender systems and report a consistent improvement in performance.

Chapter 7

Information Filtering with Topic Modeling

7.1 Introduction

In previous chapters, we observed that topical modeling can help or improve performance in many informaiton filtering tasks in online conversational media. In this chapter, we would like to investigate this direction more thouroughly. More specifically, we would like to take Twitter as a typical example of online conversational media to study how topic modeling can enhance the performance.

In recent years, social networks such as Facebook, Myspace and Twitter have become important communication tools for people across the globe. These websites are increasingly used for communicating breaking news, eyewitness accounts and organizing large groups of people. Users of these websites have become accustomed to receiving timely updates on important events, both of personal and global value. For instance, Twitter was used to propagate information in real-time in many crisis situations such as the aftermath of the Iran election, the tsunami in Samoa and the Haiti earthquakes. Many organizations

and celebrities use their Twitter accounts to connect to customers and fans.

Recent studies in a variety of research areas show increasing interests in micro-blogging services, especially Twitter. Early work mainly focused on quantitative studies on a number of aspects and characteristics of Twitter. For example, Java et al. [104] studied the topological and geographical properties of Twitter's social network in 2007 and found that the network has high degree correlation and reciprocity, indicating close mutual acquaintances among users. Krishnamurthy et al. [123] studied the geographical distribution of Twitter users and their behaviors among several independent crawls. The authors mostly agree with the classification of user intentions presented by Java et al., but also point out evangelists and miscreants (spammers) that are looking to follow anyone. Weng et al. [207] studied the problem of identifying influential users on Twitter by proposing an extension of the PageRank algorithm to measure the influence taking both the topical similarity between users and the link structure into account. They also presented evidence to support the existence of homophily in Twitter. In their work, they utilized topic models (described below) to understand users' interests.

Among the research mentioned above and others, researchers wish to use messages posted by users to infer users' interests, model social relationships, track news stories and identify emerging topics. However, several natural limitations of messages prevent some standard text mining tools to be employed with their full potentials. First, messages on Twitter (which are called "tweets") are restricted to 140 characters. This is substantially different from traditional information retrieval and web search. Second, within this short length, users invented many techniques to expand the semantics that are carried out by the messages. For example, when posting external URLs, users may use URL shortening services (e.g., http://www.bit.ly). In addition, users heavily use self-defined hash tags starting with "#" to identify certain events or topics. Therefore, from the perspective of length (e.g., in characters), the content in messages is limited while it may convey rich

meanings.

Topic models [25] are powerful tools to identify latent text patterns in the content. They are applied in a wide range of areas including recent work on Twitter (e.g., [163]). Social media differs from some standard text domain (e.g., citation network, web pages) where topic models are usually utilized in a number of ways. One important fact is that there exists many "aggregation strategies" in social media that we usually want to consider them simultaneously. For example, on Twitter, we usually want to obtain topics associated with messages and their authors as well. Researchers typically only discuss one of them. Weng et al. [207] trained a topic model on aggregated users' messages while Ramage et al. [163] used a slightly modified topic model on individual messages. Neither of them mentioned the other possibility. Indeed, to our knowledge, there is no empirical or theoretical study to show which method is more effective, or whether there exists some more powerful way to train the models.

In this paper, we want to address the problem of how to effectively train a standard topic model in short text environments. Although our experiments are solely based on Twitter, we believe that some of the discussions can be also applied to other scenarios, such as chat logs, discussion boards and blog comments. More specifically, we want to answer these questions in the paper:

- If we use different aggregation strategies and train topic models, do we obtain similar topics or are the topics learned substantially different?
- Can we learn a topic model more quickly that retains its usefulness, without any modifications to standard models?
- Can we shed some light on how we can build new models to fully utilize the structure of short text environments?

With a set of carefully designed experiments in both quantitative and qualitative perspective and two more real-world classification problems, in this paper, we make the following contributions:

- Topics learned by using different aggregation strategies of the data are substantially different from each other.
- Training a standard topic model on aggregated user messages leads to a faster training process and better quality.
- Topic mixture distributions learned by topic models can be a good set of supplementary features in classification problems, significantly improving overall classification performance.

This chapter is organized as follows. In Section 7.5, we outline some related work on the topic. In Section 7.2, we introduce several methods to learn topic models on Twitter. Section 7.3 details our experiments and major conclusions. In Section 7.4, we summarize our contributions.

7.2 Methodology

In this section, we will introduce several methods to train topic models on Twitter and discuss their technical details. In this paper, we mainly consider two basic models: LDA and author-topic model [171]. We first briefly review these two models and then discuss their adaptation to Twitter.

7.2.1 LDA and the Author-Topic Model

Latent Dirichlet Allocation is an unsupervised machine learning technique which identifies latent topic information in large document collections. It uses a "bag of words" approach,

which treats each document as a vector of word counts. Each document is represented as a probability distribution over some topics, while each topic is represented as a probability distribution over a number of words. LDA defines the following generative process for each document in the collection:

1. For each document, pick a topic from its distribution over topics.

2. Sample a word from the distribution over the words associated with the chosen topic.

3. The process is repeated for all the words in the document.

More formally, each document in the collection is associated with a multinomial distribution over T topics, which is denoted as θ. Each topic is associated with a multinomial distribution over words, denoted as ϕ. Both θ and ϕ have Dirichlet prior with hyper-parameters α and β respectively. For each word in one document d, a topic z is sampled from the multinomial distribution θ associated with the document and a word w from the multinomial distribution ϕ associated with topic z is sampled consequently. This generative process is repeated N_d times where N_d is the total number of words in the document d.

The Author-Topic Model (AT model) is an extension of LDA, which was first proposed in [172] and further expanded in [171]. Under this model, each word w in a document is associated with two latent variables: an author, x and a topic, z. Similarly to LDA, each author in the collection is associated with a multinomial distribution over T topics, denoted as θ. Each topic is associated with a multinomial distribution over words, denoted as ϕ. Here, differing from LDA, the observed variables for an individual document is the set of authors and the words in the document. The formal generative process of Author-Topic Model is as follows:

1. For each document, given the vector of authors.

2. For each word in the document, conditioned on the author set a_d, choose an author $x_{di} \sim \text{Uniform}(a_d)$.

3. Conditioned on x_{di}, choose a topic z_{di}.

4. Conditioned on z_{di}, choose a word w_{di}.

Here, one important difference between the AT model and LDA is that there is no topic mixture for an individual document. Therefore, if we want to model documents and authors simultaneously, certain extension or special treatment is needed. A detailed description of the model can be found in [171].

7.2.2 Topic Modeling Schemes

Recall that our goal is to infer a topic mixture θ for both messages and authors in the corpus. In this sub-section, we will introduce several methods to achieve this goal.

First, we discuss a very natural choice of training models. The process is as follows:

1. Train LDA on all training messages.

2. Aggregate all training messages generated by the same user into a training profile for that user.

3. Aggregate all testing messages generated by the same user into a testing profile for that user.

4. Taking training user profiles, testing user profiles and testing messages as "new documents", use the trained model to infer a topic mixtures for each of them.

We denote this method as the MSG scheme. Note that we do not combine all user profiles into a single set of user profiles simply because some users may be part of the training set,

and thus the aggregation of all user profiles may give an unfair advantage to the model to achieve better performance.

We can also train the model on aggregated user profiles, which leads to the following process:

1. Train LDA on aggregated user profiles, each of which combines all training messages generated by the same user.

2. Aggregate all testing messages generated by the same user into testing user profiles.

3. Taking training messages, testing user profiles and testing messages as "new documents", use the trained model to infer a topic mixture for each of them.

We denote the method as the USER scheme.

The third scheme, which we denote as the TERM scheme, is more unusual. The process is as follows:

1. For each term in the training set, aggregate all the messages that contain this term into a training term profile

2. Train LDA on all training term profiles.

3. Build user profiles in training and testing set respectively.

4. Taking training messages, training user profiles, testing user profiles and testing messages as "new documents", use the trained model to infer a topic mixture for each of them.

The rationale for this scheme is that on Twitter, users often use self-defined hash tags (i.e., terms starting with "#") to identify certain topics or events. Building term profiles may allow us to obtain topics related to these hash tags directly.

These schemes each have their own advantages. For MSG, it is straightforward and easily understandable but the training process is based on individual messages, whose content is very limited. The model may not have enough information to learn the topic patterns. More specifically, the occurrences of terms in one message play less discriminative role compared to lengthy documents (e.g., aggregated user profiles or term profiles) where the model has enough term counts to know how terms are related. For the USER and TERM schemes, the models have enough content and might provide a more "accurate" result.

For the AT model, we extend it to allow each message to have a "fictitious" author who is indeed the message itself. Thus, for each message, we either sample the words from the author specific topic mixture or sample them from the message specific topic mixture. Note that the relationship between message specific "route" and author specific "route" is "OR". In other words, we can imagine the process is that an author is writing a message that he will mainly choose the words he is usually interested while choosing some set of words more specific to the current message. Therefore, under this assumption, most of terms in a particular message will choose author "route". This "OR" relationship indeed allows us learn a relatively accurate model for authors but less satisfied model for messages. In our experiments, we find that the topic mixture for messages learned by the extended AT model is usually too sparse and leads to worse results than the MSG scheme. In this paper, we use the AT model to denote the extended AT model with message specific mixtures.

There is another aspect of issues related to different schemes. Usually, the number of users is several magnitude less than the number of messages. Therefore, it would take significantly less time to train a model with the USER scheme rather than the MSG scheme. The same argument can be made for the TERM scheme as well. In addition, the assumption of topic mixture of topic models might eventually lead to different optimal choice of T (the number of topics) for different schemes. For the MSG scheme, we are modeling

the number of topics existing in messages. Since a message is short and the number of messages is huge, we usually need a larger number of topics to obtain a reasonable model. On the other hand, for the USER scheme, we are modeling the number of topics for users. We can arguably say that each user may only have a relatively small number of topics that they are interested in and the total number of users are comparatively smaller than the volume of messages. Hence, through our experiments, the optimal number of topics is usually smaller than its in MSG scheme.

Note that in this paper we only explore schemes that do not require any significant modifications to the LDA or AT models. We do believe that better extensions of LDA which consider authors and messages simultaneously might be more useful.

7.3 Experiments

In this section, we present the experimental evaluation of the schemes discussed in the previous section. For the experiments we use Twitter data obtained through both the streaming and normal APIs. We begin by describing some preprocessing steps of our data. Then, we test a variety of schemes discussed in the previous section on two realistic tasks. By studying the results, we will show that topic modeling is a powerful tool for short text messages.

7.3.1 Tasks

In our experiments, we have two different tasks, whose performance can be potentially enhanced by topic modeling techniques:

- Predicting popular Twitter messages
- Classifying Twitter users and corresponding messages into topical categories

For the first task, we consider the number of times a message has been retweeted as a measure of popularity. Therefore, we convert the problem into predicting whether a message will be retweeted in the **future**. Since we only have an incomplete set of Twitter messages and we cannot directly recover complete retweet patterns, we need to construct a reasonable dataset from our sample. Consider a collection of messages, some of which are duplicates of others. Before we measure if two messages are "similar", we take the following preprocessing steps: 1) We remove links from the messages; 2) We remove any word stating with the "@" character; 3) We remove non-latin characters in the message and convert all characters to lower case; and, 4) We calculate the hash value of all the messages. We use MD5 to obtain the signature for all messages. If two messages share the same MD5 value, we define them as "similar" to each other. We group similar messages together and sort them by time. All the versions of a message form a chain. For all messages in the chain except the first, we further filter out those messages without "RT". In other words, it does not matter if the first message is a retweet, but all subsequent messages in the chain must be retweets. For all filtered chains, if there are n messages in a particular chain, we take the first $n-1$ messages as "positive instances", which means they will be retweeted in the future, and the last one as "negative instance". In addition, all other messages which are not in any chains are also considered as "negative instances". Our task is to correctly predict all "positive instances" in the dataset.

The second task is more straightforward. In several Twitter directories (e.g., http://www.wefollow.com) and in the official Twitter site, lists of users with categories associated with them is provided. We take more than 250 verified users from the official Twitter Suggestions categories[1] under the assumption that these verified accounts are recognized as valid by real people and organizations. The categories do not overlap. We monitored the latest 150 messages generated by these users and try to classify the

[1] http://twitter.com/invitations/suggestions

Table 7.1: Users From Twitter Suggestions

Category ID	Category Name	# of Users
0	Art & Design	3
1	Books	3
2	Business	8
3	Charity	15
4	Entertainment	42
5	Family	4
6	Fashion	5
7	Food & Drink	19
8	Funny	23
9	Health	9
10	Music	43
11	News	16
12	Politics	27
13	Science	4
14	Sports	39
15	Technology	22

messages and the account into their corresponding categories which we obtained from Twitter Suggestion, under the assumption that these verified users strongly adhere to their corresponding categories that most of the messages generated by them are in the same topic.

Prior to attempting the two tasks, we also studied the topics learned by the models empirically mainly from two aspects: 1) Whether the topics obtained by different schemes are similar or not; and, 2) What is the quality of the topics. We compare the topics in both qualitative and quantitative ways.

7.3.2 Dataset

Our experiments employ the data from Twitter's APIs[2]. For the first task, we collected messages through Twitter's Streaming API, which is a push-style API with different levels

[2]http://dev.twitter.com/

Table 7.2: "Similar" Topics Found by JS Divergence

The Topic Obtained by MSG scheme
[link] our from help world their people news more haiti red photo every two school end american change water million learn women through visit america fight money far girls national wine save young office children giving earth month community needs local trip relief future project malaria uk ones #haiti number program college south power donate launch between worth education full others students history safe room group lives summer during california earthquake past charity
The Topic Obtained by USER scheme
[link] rt and we day on your is us help haiti are by from you new world with about this have red people support at thanks join out will more great twitter can their up water read video w check today were make work here get photo what please last be women live kids an school children who save event vote now project relief pls malaria life #haiti friends every them has watch donate team thank follow sign global text keep working thx do need free learn earthquake many community million

of access which constantly delivers a small fraction of Twitter messages over a permanent TCP connection. We were granted the "Garden-hose" level of access at that time, which the company describes as providing a "statistically significant sample" of the messages that flow through their system. In our experiments, we use messages from the first and second week of November 2009 but we also find similar results by conducting the same experiments on other weeks. In order to reduce the dataset to a reasonable size that can be used to evaluate the techniques easily, we remove all non-latin characters from the messages. In addition, we also remove the users who only appear once in our dataset, with their corresponding messages. This results in a dataset of 1,992,758 messages and 514,130 users. In our experiments, we neither remove stop words nor perform stemming on the words. We replace all URLs with the word "link" and keep all hash tags. Therefore, we have 3,697,498 distinct terms for the two weeks of data.

For the second task, we crawled 274 verified users of 16 categories from Twitter Suggestion and their last 150 messages if available. In order to classify users, we aggregate all the messages generated by the same user into a giant document, denote as a "user

profile". Similarly, we do not remove stop words and do not perform stemming. Thus, the dataset contains 52,606 distinct terms and 50,447 messages in total. The detailed number of users per category is shown in Table 7.1.

7.3.3 Evaluation Metrics & Parameters Setting

We cast both tasks into classification problems where the first one is to classify messages into retweets and non-retweets (note, "retweets" represent the messages will be retweeted in the **future**) and the second is to classify messages and users into topical categories. The baseline method for both tasks is a classifier using TF-IDF weighting values as the features.

For the first task, our basic evaluation scheme is to train the classifier on the first week and test it on the second week while for the second one, a simple cross-validation scheme is used. For the first task, we use Precision, Recall and F-Measure (F1 score) as the evaluation metric with their definitions shown as follows:

$$\text{Precision} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{false positives}}$$

$$\text{Recall} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{false negatives}}$$

$$\text{F-Measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The vast majority of the instances in our dataset are negative ones (e.g., the messages will not be retweeted in the future). Therefore, a naive classifier may easily achieve more than 90% accuracy by choosing every instance as negative, which does not make much sense in our case. Hence, we do not report any results based on accuracy for the first task. For the second task, we use classification accuracy as the evaluation metric. We not only look at

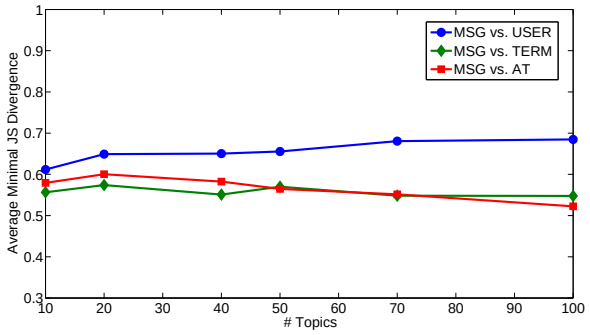

Figure 7.1: The Average Minimal JS Divergence

the classification accuracy for each category but also care about the overall classification accuracy.

Throughout the experiments, we use L2-regularized Logistic Regression as our classifier[3]. In our preliminary experiments, we also tried L1 regularization, which corresponds to learning a sparse representation of features. Since we did not find any performance gains through L1 regularization, we only report the results on the L2 regularized classifier.

All the topic models used in the experiments have symmetric Dirichlet priors. We notice that asymmetric priors may lead to better results, suggested by [194]. However, in order to reduce the effect of optimizing hyper-parameters, we fix all of them to symmetric Dirichlet priors. More specifically, for β, we set it to 0.01 in all experiments and for α, we adopt the commonly used $50/T$ heuristics where T is the number of topics. In our experiments, we use Collapsed Gibbs Sampling [79] with speed-up techniques introduced in [216], which can be scaled to our large dataset.

7.3.4 Topic Modeling

In this section, we mainly study two questions: 1) whether different training schemes cause the model to learn different topics from the dataset; and, 2) what is the quality of topics learned from the dataset by different schemes. The dataset we used in this sub-section is the topical classification dataset described in Section 4.1.

In order to answer the first question, we need to map topics learned by different schemes. Due to the "exchangeable" property of topic models [28], the topics learned from different runs of the models are not directly correspond, even for the exactly same settings. Therefore, a mapping process is required to find same or similar topics. In this work, we use Jensen-Shannon (JS) divergence to measure the similarity between topics. The JS divergence is a symmetric measure of the similarity of two pairs of distributions. The measure is 0 only for identical distributions and approaches infinity as the two differ more and more. Formally, it is defined as the average of the KL divergence of each distribution to the average of the two distributions:

$$D_{JS} = \frac{1}{2} D_{KL}(P||R) + \frac{1}{2} D_{KL}(Q||R)$$
$$R = \frac{1}{2}(P + Q)$$

where $D_{KL}(A||B)$ represents the KL divergence between variable A and B. In our case, the KL divergence is calculated as follows:

$$D_{KL}(A||B) = \sum_{n=1}^{M} \phi_{na} \log \frac{\phi_{na}}{\phi_{nb}}$$

where M is the number of distinct term types and ϕ_{na} is the probability of term n in topic a. For each topic i, we obtain a corresponding topic j with the minimal JS divergence

[3] http://www.csie.ntu.edu.tw/~cjlin/liblinear/

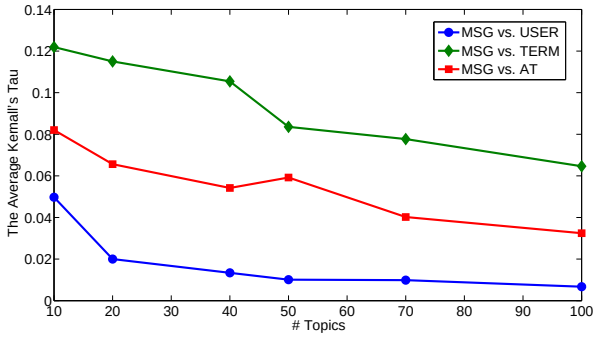

Figure 7.2: The Average Kendall's τ

score where topic i and j are trained through different schemes.

Let us first look at the results qualitatively. In Table 7.2, we list two topics identified by minimal JS divergence as "similar topics" where two models are trained on the dataset for the second task and the number of topics $T = 10$. The upper part of the table shows the topic found by the MSG scheme and the bottom part shows the topic obtained by the USER scheme. All the terms shown in the table are the topic terms sorted by ϕ scores. In other words, these terms are generated by the topics with high probabilities. Not very surprisingly, the top terms found by different schemes do not match with each other exactly. However, by carefully reviewing the terms, we find that most of them are related to some news events (e.g., Haiti earthquake) and politics.

In order to better quantify the difference between topics, we use two metrics based on JS divergence. One is to calculate the average divergence between "similar" topics, which we denote "the average minimal JS divergence". More specifically, for each topic i, we first find a "similar" topic j with minimal JS divergence. Then, we calculate the average of JS divergence over all discovered "similar" topics. Figure 7.1 displays the average minimal JS divergence between different models. In this figure, we see that there is obvious difference between topics learned by different schemes or models. Topics learned by the USER

scheme are substantially different from the topics learned by the MSG scheme and JS divergence slightly increases with increasing number of topics. Compared to the USER scheme, topics learned by the TERM scheme and the AT model are closer to the topics of the MSG scheme. Note that almost all the JS divergence values are far from 0, which indicates that the probabilities of terms in each topic indeed differ apart.

From JS divergence, we conclude that the probabilities learned are different but we do not know how these difference may influence the relative positions of terms ranked in the topics. Therefore, the second metric we use is to measure the difference between rankings of terms obtained by topics. As shown in Table 7.2, while some of the terms found by different schemes are all ranked highly (e.g., haiti, relief), the exact ranking position is not the same. By looking at the discrepancy between rankings, we can understand how topics deviate from each other and how different models agree with each other. Here, we use Kendall's τ to measure the agreement between rankings. Given two different rankings of the same m items, Kendall's τ is defined as:

$$\tau = \frac{P - Q}{P + Q}$$

where P is the number of pairs of items in two rankings that are concordant and Q is the number of pairs of items in two rankings that are not concordant. τ ranges from -1 to 1, with 1 meaning the two rankings are identical and -1 meaning one is in the reverse order of the other. If $\tau = 0$, it means that 50% of the pairs are concordant while 50% of the pairs are discordant. We take the top 500 terms ranked by "similar" models identified by minimal JS divergence and calculate the τ values. Figure 7.2 shows the results of τ values between "similar" topics. Two immediate observations can be discovered. First, the disagreement between the MSG scheme and the USER scheme is substantially larger than other schemes. Second, as the number of topics increases, the disagreement increases.

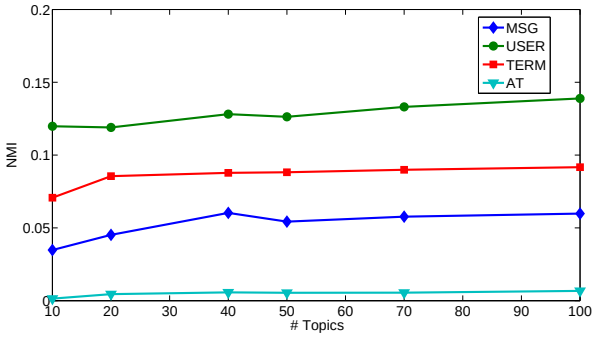

Figure 7.3: Normalized Mutual Information

Next, we would like to know the quality of topics found by the models. The dataset we used is still the topical classification dataset containing sixteen categories. Since we know the ground truth label of all the messages in the dataset (their categories), we can measure the quality by how likely the topics agree with the true category labels. Here, we use Normalized Mutual Information (NMI), which can be defined as follows:

$$\text{NMI}(\Omega, \mathbb{C}) = \frac{I(\Omega, \mathbb{C})}{[H(\Omega) + H(\mathbb{C})]/2}$$

where $I(\Omega, \mathbb{C})$ is mutual information between set Ω and $\mathbb{C}$ and $H(A)$ is the entropy. NMI is always a number between 0 and 1. NMI may achieve 1 if the clustering results can exactly match category labels while 0 if two sets are independent. Details of the calculation of NMI can be found in [139]. For each message, we use the maximum value in topic mixture θ to determine its cluster, which leads to a "hard" clustering result. After this mapping process, we compute NMI with the labels and the results are shown in Figure 7.3. From the figure, we see that NMI values are low in general. Clusters assigned by the USER scheme matches labels significantly better than other schemes. The NMI values by the AT model are nearly zero, indicating that they almost do not match class labels at all. As discussed before, the AT model does not provide a fully formalized generation process for

Table 7.3: The Comparison of Performance on Retweet Prediction

Scheme	Precision	Recall	F1
TF-IDF	0.4216	0.3999	0.4105
MSG (100)	0.5088	0.2837	0.3643
USER (40)	0.6075	0.3677	0.4581
TERM (70)	0.5292	0.3061	0.3879
AT (70)	0.4811	0.2654	0.3421
TF-IDF + MSG	0.5150	0.3546	0.4200
TF-IDF + USER	0.6142	0.3897	0.4768
TF-IDF + TERM	0.5303	0.3582	0.4276
TF-IDF + AT	0.4736	0.3622	0.4104

documents. Therefore, the quality of topic mixture learned for messages is comparatively poor.

In conclusion, topics obtained by different schemes usually vary substantially. As shown in the experiments, the USER scheme might achieve better agreement with predefined labels, if available.

7.3.5 Predicting Popular Messages

In this section, we would like to see how the schemes and models discussed can influence classification performance. Here, we consider the problem of predicting potential "retweets". Remember, we treat the problem as a classification problem where the input is a set of features and the output tells us whether the target message will be retweeted in the future or not.

We first use TF-IDF weighting scores as features and train a Logistic Regression classifier. The result is shown in the first row of Table 7.3. Then, we train topic models according to the different schemes and obtain topic mixture θ for both messages and authors as introduced in the Section 7.2. For different schemes, we only report the best

performance and its corresponding number of topics. We only test the number of topics in the range of 20 to 150. The results are shown from the second row to the fifth row (the first half of the Table) in Table 7.3. The first conclusion we can draw is that most of results are worse compared to the baseline, TF-IDF, while only the topics trained by USER scheme significantly outperform the baseline. In the last sub-section, we see that the topics trained by USER scheme achieve higher NMI value, which implies that USER scheme might more likely match the underlying category information. Although other schemes do not perform well, we notice that the Precision is improved by all these schemes. If we argue that Precision is more valuable in this task (because once we make a "positive" decision, we have less chance to be wrong), we can conclude that topic models indeed help us.

Some literature [28] suggested that if we solely use topic mixture as features, we may not achieve better performance than TF-IDF. Thus, we combine topic model features and TF-IDF features and obtain the results in the second half (from 6th row to the bottom) of the Table 7.3. The results are trained on a classifier using the best performing topic model features with TF-IDF features. We can see that most of them improve performance and TF-IDF with USER scheme outperforms the previous best one that only uses the topic features. Surprisingly, the AT model performs the worse in the experiments and combining TF-IDF features does not give the AT model much boost in the performance.

In this task, we see that although sometimes topic features may not outperform simple TF-IDF features, it is good practice to combine them. USER scheme consistently provides good results, compared to other models.

7.3.6 User & Message Classification

In this section, we will see the results of the second task, classifying messages and authors into topical categories. First, let us turn our attention to the performance on message

Table 7.4: The performance of TF-IDF features on Message Classification

Category	Accuracy	Category	Accuracy
0	0.3000	1	0.2143
2	0.2756	3	0.5909
4	0.4722	5	0.1250
6	0.2577	7	0.3553
8	0.3459	9	0.6471
10	0.5544	11	0.4026
12	0.5350	13	0.3553
14	0.6220	15	0.4185
		Average:	0.4792

Table 7.5: The best performance of USER Scheme on Message Classification

Category	Accuracy	Category	Accuracy
0	0.5000	1	0.0000
2	0.5128	3	0.9583
4	0.8223	5	0.0000
6	0.3814	7	0.8899
8	0.9082	9	0.7386
10	0.8718	11	0.8636
12	0.8132	13	0.5263
14	0.9330	15	0.9022
		Average:	0.8291

classification. Recall that we have 274 users from 16 categories in the dataset. For each user, we assume that all the messages generated by this user fall into the same category as the user. Therefore, for message classification, we use 90% of messages for training and 10% for testing and report the results on 5-fold cross validation. The baseline method is to use the TF-IDF weighting scores as features to train the classifier, which is shown in Table 7.4. Note that the category ids correspond to the categories introduced in Table 7.1. The overall accuracy is around 47% where the high performance is achieved in "Health" and "Sports" categories.

Again, similar to the first task, we use the topic mixture θ for both messages and

Table 7.6: The best performance of MSG Scheme on Message Classification

Category	Accuracy	Category	Accuracy
0	0.5000	1	0.3036
2	0.1218	3	0.9583
4	0.6934	5	0.0000
6	0.1753	7	0.8899
8	0.8894	9	0.8693
10	0.8277	11	0.7403
12	0.7749	13	0.5263
14	0.9732	15	0.8451
		Average:	0.7838

users learned by topic models as features. We test the features in two settings, only using topic features and combining with TF-IDF features. We only report the best performance with its number of topics while we test the topic numbers from 10 to 150. Table 7.5 shows the best results obtained by USER scheme when the number of topics $T = 50$. Note, the overall accuracy is significantly improved and it is almost twice as accurate as raw TF-IDF features. However, we also note that the classifier results in zero accuracy in some categories. Category 1 ("Books") and category 5 ("Family") are two cases where the classifier does not achieve one valid instance. One potential reason for this phenomenon is that the number of instances in these categories are significantly smaller than other categories, which prevent the classifier and topic models to learn enough information about them. Table 7.6 shows the best results by the MSG scheme as $T = 100$. First, the overall accuracy is improved by TF-IDF features but lower than USER scheme. Second, we still have "Family" category with 0 accuracy. Due to space limits on the paper, we do not include detailed performance results for the TERM scheme and the AT model. The highest accuracy achieved by the TERM scheme is 0.6684 with $T = 100$ and by the AT model is 0.5459 when $T = 150$. Both of them are far worse than the MSG and USER schemes but still better than raw TF-IDF scores. When we combine topic features with

Table 7.7: The best performance of TF-IDF + USER on Message Classification

Category	Accuracy	Category	Accuracy
0	0.3000	1	0.2500
2	0.2692	3	0.5985
4	0.4776	5	0.1250
6	0.2680	7	0.3491
8	0.3388	9	0.6797
10	0.5492	11	0.4026
12	0.5478	13	0.3816
14	0.6327	15	0.4266
		Average:	0.4838

TF-IDF features, unlike the first task shown in the last sub-section, the performance is always worse than only using topic features and only slightly better than solely using TF-IDF values. We only report the best results in Table 7.7, which is trained through USER scheme with $T = 40$. We notice that by combining TF-IDF features we can avoid the "zero" accuracy situation in all our experiments. Therefore, to some extent, TF-IDF features can capture some micro-level characteristics of categories while the topic features are usually too high level (since the feature is indeed topic mixture not the topic distribution itself).

Now, let us turn to the problem of classifying users into topical categories. Similar as message classification, we split 90% of messages and aggregate the messages in training set for each user to build the user profiles. So, the training user profiles and testing profiles are always different and do not mixed. Again, TF-IDF is calculated as features for user profiles, which are aggregations of all messages generated by the same user. The baseline is shown in Table 7.8. Surprisingly, the performance is very high, almost twice higher than the baseline in message classification. For category "Business" and "Charity", the classifier distinguished all instances successfully. In fact, in our experiments, the classifier trained on topic features performs much worse than the baseline regardless of schemes.

Table 7.8: The Performance of TF-IDF on User Classification

Category	Accuracy	Category	Accuracy
0	0.5000	1	0.6667
2	1.0000	3	1.0000
4	0.9756	5	0.5000
6	0.4000	7	0.7895
8	0.8261	9	0.8750
10	0.9767	11	0.8750
12	1.0000	13	0.5000
14	0.9474	15	0.8636
		Average:	0.9051

We only report the best performing results in Table 7.9, which is obtained through USER scheme with $T = 20$. We notice that not only the overall accuracy is not as good as TF-IDF features but using topic features also results in several zero accuracy in different categories. One reason is again the content in those categories is limited. An interesting point is that if we combine TF-IDF features with topic features, the overall performance is still around 90% (in fact, only with marginal improvement). Remember, for user profiles, we crawled the latest 150 updates for each user, if available. Therefore, for most users, the profile already contain enough information to learn. This situation is significantly different from message classification where we have the problem of sparsity.

Compared to the results on message classification where topic features play an important role to improve the performance and user classification where topic features fail to outperform the baseline, we believe that topic models can help us model short text while for longer content, more sophisticated models might be required to improve performance (e.g., Supervised LDA [27], Label LDA [164]).

Table 7.9: The Best Performance of USER on User Classification

Category	Accuracy	Category	Accuracy
0	0.0000	1	0.0000
2	0.0000	3	0.5333
4	0.5610	5	0.0000
6	0.0000	7	0.1053
8	0.0000	9	0.5000
10	0.6279	11	0.0000
12	0.7600	13	0.0000
14	0.7895	15	0.3182
		Average:	0.4380

7.4 Summary

Although we do not introduce new topic models to address the issues of short text modeling especially in microblogging environments in this paper, our work sheds some light on how research on topic models can be conducted for short text scenarios. More specifically, through our experiments, we demonstrate that the effectiveness of trained topic models can be highly influenced by the length of the "documents"; namely, a better model can be trained by aggregating short messages. This argument has attracted little attention in the research community in the past and should be justified through more thorough experiments and theoretical analysis. In addition, our empirical study demonstrated that topic modeling approaches can be very useful for short text either as solely used features or as complementary features for multiple real-world tasks. (Note that this does not mean that the model itself should be trained on short text and we show that a model trained on aggregated longer text can yield better performance.) We also showed that when content information is already large enough (e.g., in user classification), topic models become less effective compared to simple TF-IDF scores. Moreover, through the experiments, we showed that the simple extension to the AT model does not yield better modeling for messages and users and indeed it is worse than training a standard LDA model on user

aggregated profiles. We conjecture that the reason may be the "OR" nature of the AT model while a message is either "generated" by the message or by an author. We suggest that future models might examine how to model a hierarchical structure between users and messages.

In this paper, we conducted extensive qualitative and quantitative experiments on three proposed schemes based on standard LDA and one extended model based on the AT model. We compared a number of aspects of these schemes and models, including how the topics learned by these models differ from each other and their quality. In addition, we showed how topic models can help other applications, such as classification problems. In the experiments we demonstrated that topic models learned from aggregated messages by the same user may lead to superior performance in classification problems and topic model features can improve performance in general, especially when the research targets are messages.

7.5 Bibliographic Notes

Topic modeling is gaining increasingly attention in different text mining communities. latent Dirichlet allocation (LDA) [28] is becoming a standard tool in topic modeling. As a result, LDA has been extended in a variety of ways, and in particular for social networks and social media, a number of extensions to LDA have been proposed. For example, Chang et al. [41] proposed a novel probabilistic topic model to analyze text corpora and infer descriptions of the entities and of relationships between those entities on Wikipedia. McCallum et al. [141] proposed a model to simultaneously discover groups among the entities and topics among the corresponding text. Zhang et al. [223] introduced a model to incorporate LDA into a community detection process. Similar work can be found in [135] and [147]

Related to this work, where we need to obtain topic mixture for both messages and authors, Rosen-Zvi et al. [172] introduced an author-topic model, which can flexibly model authors and their corresponding topic distributions. In their experiments, they found that the model outperforms LDA when only small number of words are observed in the test documents. Ramage et al. [164, 163] extended LDA to a supervised form and studied its application in micro-blogging environment. Phan et al. [159] studied the problem of modeling short text through LDA. However, their work mainly focused on how to apply it to Wikipedia and they did not provide any discussion on if there is other ways to train a same model.

In web search, this line of research usually employs search engines directly. For example, Sahami et al. [174] introduced a kernel function based on search engine results. Yih et al. [218] further extended the method by exploiting some machine learning techniques.

Chapter 8

Topic Modeling: Multiple Text Streams with Temporal Dynamics

8.1 Introduction

In the previous chapter, we have demonstrated that topic modeling can help content understanding in online conversational media for several tasks. From this chapter, we enter the second part of this dissertation, which is to explore how we design specific topic models to consider a number of important aspects of online conversational media, such as dynamic temporal data and geographical tagged data.

Social-networking tools such as Facebook, LinkedIn and Twitter, have become the communication tools of choice for a large number of online users. Such tools are increasingly used for disseminating breaking news and eyewitness accounts, and even for organizing flash mobs and protest groups. For instance, Twitter was heavily used in a number of international events, such as the Iran election in 2009, the Haiti earthquakes in 2010, and the tsunami in Japan in 2011. More recently, social networking services were instrumental in facilitating the political upheavals in the Middle East. Social media as

well as the on-line publishing of more established media (e.g., newspapers, magazines and televison) have attracted a lot of attention from both researchers and product developers.

This increasing use of social media has resulted in a refocusing of research activities onto related problems, many of which are new. For example, there exists an argument as to whether social media have influenced traditional media sources and in what sense, or vice versa. In addition, people are wondering whether the topics that are shared and discussed on social media significantly differ from traditional information sources and how these topics are transferred from one source to another. Moreover, questions about the differences between various types of social media (e.g., blogs, community-based questions-and-answer portals and microblogging services) have been raised continuously both in research communities and industry. Effectively addressing these issues requires the ability to analyze multiple types of information sources over time.

Problems similar to these have been attacked from various perspectives. For modeling the temporal dynamics of information Kleinberg et al. [116, 117] proposed methods to track the volume of a single term over time. Their later work (e.g., [130]) attempts to monitor the temporal dynamics of "memes" by which the authors mean sentence fragments representing concepts. In addition work has been done to study the dynamics of blogs [78], of online knowledge sharing communities [8], of news articles and stories [130], and of microblog services [104]. While most of the above-mentioned works focused on a single media source, some authors [222, 198, 199] modified Probabilistic Latent Semantic Analysis (PLSA) [88] to simultaneously model documents from different text streams. There is also some recent work in comparing social and traditional media. Zhao et al. [226] tried to obtain latent topics from Twitter and New York Times (NYT) news articles by using topic models. Two different topic models were used to learn the topics from the two sources separately and heuristics were then applied to obtain both common and local topics. Attempts have been made to extend topic models to incorporate temporal dynamics

and topic evolution (e.g., [26, 201]). In addition to research projects, commercial products also provide tools to search and browse the dynamics of queries[1], news articles and web traffic[2], and microblogging updates[3].

While existing research offers different methods to monitor and track correlated information sources over time, many of the proposed approaches suffer from significant drawbacks. For instance most of the work on tracking information sources primarily focuses on only one type of source. Given the multiplicity of media channels however, it is potentially more useful to understand multiple information sources simultaneously. Also, tracking a single word or a meme can be quite limiting. Further, most models that consider multiple text collections either have model parameters requiring manual adjustment or have theoretical limitations (see our discussion in Section 8.6). In addition temporal factors are either not incorporated in the models or are heuristically embedded. For temporal topic models most approaches adopt a Markovian assumption that may not be suitable for social media. Indeed, none of them utilize recent research findings of temporal variations of information in social media [130, 213].

In this chapter we address the problem of modeling multiple text streams, including their temporal dynamics, in a principled manner. Our work builds on recent work in both information dynamics and topic models. More specifically, we extend topic models by allowing each text stream to have both local and shared topics. For temporal modeling, we associate each topic with a time-dependent function that characterizes its popularity over time. By combining the two models, we effectively model temporal dynamics of multiple correlated text streams in a unified framework. To summarize the contributions of this chapter, the work we describe includes:

- a topic model that discovers common and uncommon topics from multiple text

[1] http://www.google.com/insights/search/
[2] http://www.google.com/trends
[3] http://www.google.com

collections

- a temporal model that characterizes the dynamic of topics over time
- a simple and potentially scalable algorithm for mining temporal topics
- interesting results from Yahoo! News and Twitter obtained by applying our model.

The remainder of this chapter is organized as follows. Section 8.6 provides the background and related work. In Section 8.2 and Section 8.3, we discuss our model in detail. Section 8.4 provides experimental results on real-world datasets. We summarize the chapter with Section 8.5, which discusses both conclusions and future work.

8.2 Correlated Text Streams

8.2.1 Model Description

Our correlated-text-stream model (`Collection Model`) is an extension of Latent Dirichlet Allocation [28] (LDA). In our `Collection Model`, we have a set S of n text streams. Associated with each stream $s \in S$ is a set T_s of *local* topics and associated with all streams is a set T_c of *common* topics. Thus the total number of topics in the model is $(\sum_s |T_s|) + |T_c|$. As in LDA, each topic k is defined as a multinomial distribution over a fixed vocabulary V, denoted as ϕ_k. Local topics $\phi^{(s)}$ are drawn from stream-dependent Dirichlet distributions $\text{Dir}(\beta^{(c)})$ while common topics $\phi^{(c)}$ are drawn from a stream-independent Dirichlet distribution $\text{Dir}(\beta^{(c)})$. Each document d in a stream s, has an associated Bernoulli distribution with parameter $\eta_{d,s} \sim \text{Beta}(\gamma_s^{(s)}, \gamma_s^{(c)})$, indicating how likely the document is to choose local rather than common topics. For convenience we let $\eta_{d,c}$ (where $\eta_{d,c} = 1 - \eta_{d,s}$) represent how likely a document d is to choose common topics. The random variable $x_{d,i} \sim \text{Bernoulli}(\eta_{d,s})$ takes on one of the two values "local" or "common" for each word position i in document d. In addition, each document has two

multinomial distributions with parameter vectors $\theta_d^{(s)} \sim \text{Dir}(\alpha_s)$ and $\theta_d^{(c)} \sim \text{Dir}(\alpha_c)$ over T_s and T_c respectively, where α_s and α_c represent the two Dirichlet parameter vectors. The document generation process associated with this model is as follows:

1. For all common topics T_c, draw $\phi^{(c)} \sim \text{Dir}(\beta^{(c)})$
2. For a particular stream s
   (a) For all local topics T_s, draw $\phi^{(s)} \sim \text{Dir}(\beta^{(s)})$
   (b) For each document d in s
      i. Draw Bernoulli parameter $\eta_{s,d} \sim \text{Beta}(\gamma_s^{(s)}, \gamma_s^{(c)})$
      ii. Draw $\theta_d^{(s)} \sim \text{Dir}(\alpha_s)$
      iii. Draw $\theta_d^{(c)} \sim \text{Dir}(\alpha_c)$

      For each word position i in document d

      A. Draw $x_{di} \sim \text{Bernoulli}(\eta_{s,d})$
      B. Draw a topic $z_{di} \sim \text{Multinomial}(\theta_d^{(x_{di})})$
      C. Draw a word $w_{di} \sim \text{Multinomial}\ (\phi_{z_{di}}^{(x_{di})})$

Under this generation scheme, the probability a term w is generated by a document d is:

$$p(w|d) = \eta_{d,s}\Big(\sum_{z \in T_s} \phi_{z,w}\theta_{d,z}^{(s)}\Big) + (1 - \eta_{d,s})\Big(\sum_{z \in T_c} \phi_{z,w}\theta_{d,z}^{(c)}\Big)$$

8.2.2 Inference via Collapsed Gibbs Sampling

In order to estimate the hidden parameters in the model, we apply collapsed Gibbs sampling using the following updating rules:

$$
\begin{aligned}
&p(x_{di} = s, z_{di} = t) \propto \\
&\frac{c_{d,s-i} + \gamma_s^{(s)}}{N_d + \gamma_s^{(s)} + \gamma_s^{(c)} - 1} \frac{m_{d,z-i} + \alpha_z}{\sum_{z \in T_s} m_{d,z-i} + \alpha_z} \frac{n_{z,w-i} + \beta_w^{(s)}}{\sum_w^V n_{z,w-i} + \beta_w^{(s)}} \\
&p(x_{di} = c, z_{di} = t) \propto \\
&\frac{c_{d,c-i} + \gamma_s^{(c)}}{N_d + \gamma_s^{(s)} + \gamma_s^{(c)} - 1} \frac{m_{d,z-i} + \alpha_z}{\sum_{z \in T_c} m_{d,z-i} + \alpha_z} \frac{n_{z,w-i} + \beta_w^{(c)}}{\sum_w^V n_{z,w-i} + \beta_w^{(c)}}
\end{aligned} \tag{8.1}
$$

where $c_{d,s-i}$ is the number of words in document d assigned to local topics (excluding w_{di}), $m_{d,z-i}$ is the number of words in document d assigned to topic z (excluding the current one) and $n_{z,w-i}$ is the number of occurrences of term w assigned to topic z (excluding the current one). By using the samples from Gibbs sampling, parameters $\{\theta_d^{(s)}, \theta_d^{(c)}\}$, $\{\phi_s, \phi_c\}$ and $\{\eta_{d,s}, \eta_{d,c}\}$ can be effectively estimated as follows:

$$\theta_{d,z}^{(x)} = \frac{m_{d,z} + \alpha_z}{\sum_{z \in T_x} m_{d,z} + \alpha_z} \quad , x \in \{s, c\} \tag{8.2}$$

$$\phi_{z,w}^{(x)} = \frac{n_{z,w} + \beta_w}{\sum_{z \in T_x} n_{z,w} + \beta_w} \quad , x \in \{s, c\} \tag{8.3}$$

$$\eta_{d,x} = \frac{c_{d,x} + \gamma_s^{(x)}}{N_d + \gamma_s^{(s)} + \gamma_s^{(c)}} \quad , x \in \{s, c\} \tag{8.4}$$

The formalism of our model resembles in spirit that of Chemudugunta et al. [43] where each term is "split" into corpus-level "background" topics, document-level "special" topics and normal topics. However, their work is only for a single corpus while our model fits multiple collections. Hyper-parameters like β, α and γ can be estimated using standard methods introduced by Minka [144].

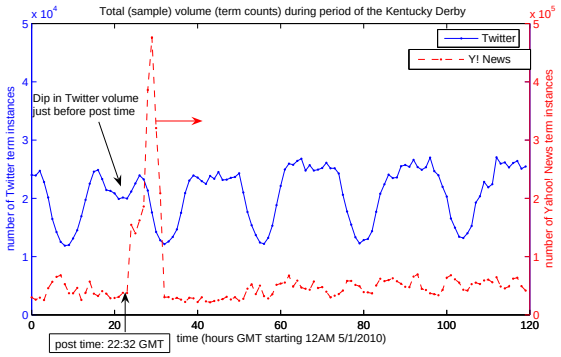

Figure 8.1: The total volume of Twitter and Yahoo! News

8.3 Modeling Temporal Dynamics

8.3.1 Temporal Dynamics for Topics

In this section we review a temporal model for news articles, introduced in [130] and present an alternate derivation. Before proceeding however, it bears pointing out that, as stated in [130], "rigorous analysis of the proposed model appears to be quite complex." The referred-to model embodies two driving forces for news-article publishing which the authors refer to as *imitation* and *recency*. The authors assert that this pair constitutes a minimum set for the purpose of explaining the temporal dynamics $n(t)$ of news-article publishing, but that the real situation is undoubtedly more complicated. We agree that there are factors beyond just these two. For example consider the Twitter and Yahoo! News total-volume data plotted in Figure 8.1 as functions of time over the first 120 hours (GMT) of May 2010. Note the enormous surge in the volume of news articles beginning after the Kentucky Derby[4], the premier American thoroughbred-horse race, and continuing for several hours. This clearly demonstrates significant elasticity in the volume capacity of the various sources contributing to Yahoo! News.

We start by assuming the following setting, which is a response that looks like a

[4] `http://en.wikipedia.org/wiki/Kentucky_Derby`

proportional controller[5] except that the "control point" $n_{\max}$ is not a constant.

$$\frac{dn}{dt} \;=\; \lambda\,[\,n_{\max} \;-\; n(t)\,], \tag{8.5}$$

where $n_{\max}$ is a function of both t and n. The form (8.5) captures the *saturation* effect mentioned above. The saturation value $n_{\max}$ varies with both n and t, however. We assume that it is the product of a term $(\zeta n(t))$ embodying the *imitation* effect mentioned above and a term (νt^{-1}) embodying the *recency* effect, where ζ and ν are adjustable parameters. Substituting the resulting expression for $n_{\max}$ into (8.5), we obtain:

$$\frac{dn}{dt} \;=\; \lambda\, n(t)\,[\,\zeta\nu t^{-1} \;-\; 1\,] \tag{8.6}$$

Next we solve this differential equation, assuming that the event occurs at $t = 0$ for convenience. For an event occurring at time t_0 let $t \to t - t_0$. We must also ensure that our solution satisfies the following boundary conditions.

1. $n(t) \;=\; 0$ for $t \leq 0$.

2. $n(t) \;\geq\; 0$ for $t > 0$.

3. $n(t) \;\to\; 0$ as $t \to \infty$.

[5] See `http://en.wikipedia.org/wiki/Proportional_control`.

The solution of (8.6) proceeds in the following steps.

$$\begin{aligned}
\frac{1}{n}\frac{dn}{dt} &= \lambda\,[\,\zeta\nu t^{-1} \;-\; 1\,], \\
\int_1^t \frac{1}{n}\frac{dn}{dt}\,dt &= \lambda\zeta\nu \int_1^t t^{-1}\,dt \;-\; \lambda \int_1^t dt, \\
\ln n(t) &= \ln n(t=1) \;+\; \lambda\zeta\nu \ln t \;-\; \lambda t \;+\; \lambda, \\
\ln n(t) &= \ln A \;+\; q \ln t \;-\; \lambda t, \\
n(t) &= A\,t^q\,e^{-\lambda t}, \qquad (8.7)
\end{aligned}$$

where $A := n(t=1)\,e^{\lambda}$ and $q := \lambda\zeta\nu$. Next we apply our boundary conditions to the solution given in (8.7). First, to enforce condition 1 we multiply the solution of (8.7) by the Heaviside unit step function $u(t)$, which equals 0 for $t < 0$ and 1 for $t > 0$. Thus, we have

$$n(t) \;=\; u(t)\,A\,t^q\,e^{-\lambda t}. \qquad (8.8)$$

Condition 2 requires that $A > 0$ and Condition 3 requires that $\lambda > 0$. The form of (8.8) has been demonstrated to capture spikes of news articles and social-media blogs [130, 213].

8.3.2 Incorporating Temporal Dynamics

In this section we describe how to incorporate the temporal model described above into our `Collection Model` and then introduce the inference approach to estimate the parameters in the model. We assume that the temporal dynamics of each topic are independent of each other. In other words, the popularity of one topic does not affect that of the other topics. We realize that this is a simplified assumption. The basic intuition behind embeding temporal dynamics into the model is to allow certain topics to have a higher probability of being selected. For example, during the Soccer World Cup in June and July of 2010, news articles and Twitter messages may naturally be more likely to

Figure 8.2: Overall Algorithm

Initialize Gibbs Sampler
while Not Converging **do**
  E-step
  For all documents in all text streams, update topic assignments using (8.1)
  M-step
  Update α, β and γ values through the method introduced in [144]
  for Each local and common topic **do**
    1) Fit gaussian function to α values
    2) Fit "temporal gamma" function by using the parameters from the previous step

    3) Re-calculate α values for topic k by using fitted function
  end for
end while

talk about the World Cup, rather than politics. We encode this notion by associating the Dirichlet parameters for each topic with a time-dependent function. This function governs the variation of those parameters and thus indirectly controls the popularity of the associated topics. More specifically, for all common topics (with parameters α_c) and local topics (with parameters α_s), we let each dimension α_k in Dirichlet parameters α to be associated with the following time-dependent function.

$$\alpha_k(t) = f_k(t) = A_k t^{q_k} e^{-\lambda_k t}$$

where $f_k(t)$ is the temporal model described in Section 8.3.1. However, if we naïvely associate α_k with f_k, the model may face difficult problems since the temporal model unrealistically assumes that the starting point of the time t for all topics is time stamp 0. In other words, different topics should have different starting times t_0. Thus, we modify it into the following form:

$$f_k(t) = C_k + u(t - t_0^k)(A_k|t - t_0^k|^{q_k} \exp(-\lambda_k|t - t_0^k|)) \tag{8.9}$$

where t_0 is the starting time stamp of the topic, A_k controls the height of the prior knowledge, q_k indicates how quickly the topic would rise to the peak, λ_k controls the rate of decay and C_k is the "noise" level of the topic. We refer to the right hand side of (8.9) as the "temporal gamma function".

The absolute-value function guarantees that the time-dependent part is only active when t is larger than t_0. Additionally, $u(t - t_0)$ is a step function that is 1 for $t \geq t_0$ and 0 otherwise. In our implementation, a "soft" version of the step function as $u(t - t_0^{(k)}) = 1/(1 + \exp(-(t - t_0^{(k)}))$ is used. Intuitively, this equation states that the prior knowledge of each topic is fixed over time (by the "noise" level C_k) until a starting point t_0 and from that point on it follows a temporal gamma function controlled by three parameters, A_k, q_k and λ_k. The crux of the problem is to estimate the values of these five parameters from the data. Note that a similar model which uses a Gaussian function to model prior knowledge was proposed in [140].

The absolute-value function and the parameter t_0 in (8.9) present challenges to model fitting (parameter estimation). To address this, rather than directly fitting $f_k(t)$, we use the following heuristic similar to that used in [140]. We first fit the following Gaussian function:

$$\alpha_k(t) \approx g_k(t) = C_k^{'} + A_k^{'} \exp(-(t - \mu_k)^2/2\sigma_k^2), \tag{8.10}$$

where μ_k is the mean and σ_k^2 is the variance. The resulting parameter values are then used to obtain initial parameter values for fitting the temporal gamma function of (8.9). This Gaussian function is straightforward to fit and its symmetric form allows us to obtain t_0 easily. We set the initial values of C_k and A_k in (8.9) to those obtained by fitting the Gaussian function and we fix $t_0^{(k)} = \mu_k - \sigma_k$. This process simplifies our inference algorithm. Note that the Gaussian approximation is only used to find initial values of the

parameters including t_0. In our later experiments we find that this approximation gives reasonable initial values.

The outline of our inference algorithm is shown in Figure 8.2. Overall, we incorporate the functional optimization problem with Gibbs sampling into a stochastic EM framework (e.g., similar to [61]). In the E-step we gather topic assignments and useful counts by Gibbs sampling through (8.1). In the M-step we optimize the proposed objective functions to obtain the updated hyper-parameters for the next iteration. More specifically, the first step is to estimate the Dirichlet parameters α from counts obtained from Gibbs Sampling. This can be done in several ways [144]. We use Newton's method in this step. The second step is to use these α values to fit the Gaussian function (8.10) and then, using the parameters from the fitted Gaussian function as initial values, to fit our temporal gamma function (8.9). For both problems we minimize the following objective functions:

$$\underset{\boldsymbol{g_k}}{\operatorname{argmin}} G_k = \frac{1}{2}\sum_t \Big(\alpha_k(t) - g_k(t)\Big)^2 \tag{8.11}$$

$$\underset{\boldsymbol{f_k}}{\operatorname{argmin}} F_k = \frac{1}{2}\sum_t \Big(\alpha_k(t) - f_k(t)\Big)^2 \tag{8.12}$$

We use the L-BFGS algorithm[136] implemented in GNU/GSL Library[73], which only requires the first-order gradients to obtain the optimal values for the parameters in both functions. Note that the method proposed here is potentially scalable to very large datasets. For example LDA-style Gibbs sampling has been scaled to very large dataset sizes by [150], which is particularly useful for our E-step. For our M-step a stochastic gradient descent can be used instead of the usual Newton's method. We denote the whole algorithm as the `Temporal Collection` model.

Table 8.1: Example Topics from Our Dataset

Comparison of Top Ranked Common Topics between `LDA` (Top) and `Temporal Collection` (Bottom)	
Title	**Top Terms**
"finance"	percent billion bank market greece financial banks debt
"crime"	police car times vehicle found york square street bomb
"junk"	link cont via #jobs #fb album super live wii #tcot #news
"oil spill'	oil gulf spill coast mexico gas drilling sea water
"junk"	dont people cant thats youre bad look tell talk
Title	**Top Terms**
"finance"	percent billion bank greece financial debt banks euro crisis
"oil spill"	oil gulf spill coast drilling mexico water louisiana
"world cup"	world cup team league final players south season club
"health care"	health medical care cancer hospital patients study research
"UK election"	minister party prime cameron political leader president
Comparison of Local Topics between News (Top) and Twitter (Bottom)	
Title	**Top Terms**
"crime"	police car times vehicle found york square street
"US election"	election party law president vote political campaign
"China"	minister china south india north chinese korea indian
"jobs"	budget tax million money pay bill federal increase cuts
"education"	school students schools board education district college
Title	**Top Terms**
"social media"	blog video post check news via twitter online facebook
"hash tags"	#fb info #quote #fail #ge #lol #ff #twibbon cont
"non-English"	les pas pour sur une cest est qui avec bien suis tout faire
"junk"	cant this wait watch next believe gonna watching just
"junk"	that would have could never were wish there

8.4 Evaluation

We utilize a real-world dataset consisting of Yahoo! News and Twitter messages from May 2010 to evaluate our method. Since the original dataset is quite large, we sample news articles and Tweets proportional to the total volume of each hour in May, resulting in 233,488 news articles and 1,736,350 Twitter messages in total. We use each hour as a time unit, which starts from 0, the first hour of May 1, 2010 to 720, the last hour of May 30, 2010. All the experiments are based on this dataset. The models used are (1) Latent Dirichlet Allocation (`LDA`), (2) Correlated Stream Model (`Collection`), introduced in Section 8.2, (3) Temporal Dynamics Topic Model (`Temporal`), introduced in Section 8.3.2 but ignoring multiple collection effects, and (4) Correlated Collection Model with Temporal Dynamics (`Temporal Collection`), introduced in Section 8.3.2. For `Collection` and `Temporal`

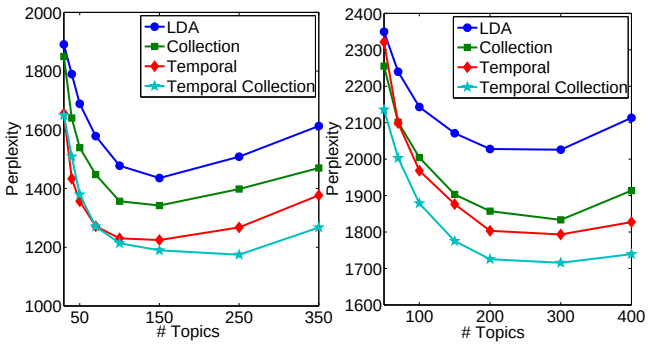

Figure 8.3: Perplexity Comparison Between Multiple Models

`Collection`, we set the number of common topics to 20 $\sim$ 50 (depending on the total number of topics) and equally divide the remaining topics into all other streams, as local topics. We do not compare with other similar methods because `Collection` and `Temporal Collection` can essentially represent the two major directions of previous work discussed in Section 8.6.

8.4.1 Perplexity Evaluation

Following common practice for comparing topic models, we use *perplexity* of the held-out test data as our goodness-of-fit measure. Perplexity is defined as

$$\exp\Big(- \frac{\sum_{d=1}^{D} \sum_{i=1}^{N_d} \log p(w_{d,i}|\mathrm{M})}{\sum_{d=1}^{D} N_d} \Big)$$

where $w_{d,i}$ represents the i^{th} term in document d, M is the model and N_d is the number of words in document d. First, we randomly sample 80% of the data as the training data and use the remaining 20% as the test data. Although this is a common evaluation procedure for topic models, it may not reflect real-world scenarios for temporal text collections because it may give additional undesirable advantages to models knowing the "future." All models are trained on the same training set and evaluated using the same test set.

In the training phase we obtain topic distributions $\boldsymbol{\phi}$ and all other hyper-parameters. In the testing phase we fix them and perform 100 Gibbs-sampling iterations for each document in the test set, obtaining θ_d. Using these newly estimated θ_d, we calculate $p(w_{d,i}|\mathrm{M}) = \sum_z^K \phi_{k,w}\theta_{d,k}$ and then compute perplexity. The result is shown on the left-hand side in Figure 8.3 where the left-hand plot shows a random 80%/20% train/test split while the right-hand plotshows a past/future 20-days/10-days train/test split. The second setting we choose is closer to real-world scenarios. We train all models on the first 20 days in May and test the perplexity on the remaining 10 days, shown on the right-hand side in Figure 8.3. As is evident in the figure, the perplexity exhibits a minimum with respect to the number of topics in both settings. As the number of topics is increased beyond that minimum, overfitting appears to set in, as was also observed in [79]. For both settings `Temporal Collection` significantly outperforms the others.

8.4.2 Common Topics and Local Topics

Here we manually compare the topics obtained by `Temporal Collection` and by `LDA` to determine which topics are meaningful and to see if any interesting patterns are discovered by the model. As we described previously, the advantage of `Temporal Collection` is to identify common topics among multiple text collections in a principled manner. Since `LDA` does not provide any mechanism for retrieving common topics explicitly, we use the following heuristic ranking method to indicate the prevalence of a topic T on both News and Twitter:

$$\frac{1}{2}\Big[\frac{n(z_T, \mathrm{News})}{\sum_{T'} n(z_{T'}, \mathrm{News})} + \frac{n(z_T, \mathrm{Twitter})}{\sum_{T'} n(z_{T'}, \mathrm{Twitter})}\Big]$$

where $n(z_T, \mathrm{News})$ is the number of tokens assigned to topic T in News and $n(z_T, \mathrm{Twitter})$ is the number of tokens assigned to the same topic in Twitter. Basically, this simple

heuristic measures how likely a topic is to be assigned to a token in both News and Twitter on average. The higher this value is, the more likely this topic will appear, on average. We rank all the topics obtained by `LDA` through this method and show the top 5 on the left top part of Table 8.1. For `Temporal Collection`, since common topics are identified automatically, we just need to rank all common topics and extract the top ones, by the following criterion: $\frac{1}{2}\Big(\mathbb{E}[\theta_i^{\text{News}}] + \mathbb{E}[\theta_i^{\text{Twitter}}]\Big)$ where $\mathbb{E}[\theta_i^{\text{News}}]$ is the expected value of θ_i for common topic t_i on news and similarly for Twitter. This equation can be interpreted as the average of the expected value of topic k appearing in a document on both collections. The quantity $\mathbb{E}[\theta_i]$ can be easily computed by $\frac{\alpha_i}{\sum_k \alpha_k}$ and normalized across all time epochs. The top 5 common topics are listed on the right top part of the same table.

The first column and the third column of the Table 8.1 show the *title* of the topics, a label given by the authors for easier interpretation. All topics (the second and the fourth column) are represented by the top ranked terms by $\phi_{z,w}$. Note that all these models are fit in an unsupervised manner in which no explicit human labels are available beforehand. From the results it is clear that both methods rank some potential common topics highly, such as "Oil Spill" and "Financial Crisis". However, it is also noticeable that simple ranking heuristics may not give appropriate scores to the topics. For instance the ranking scheme may prefer the topics from a collection that is significantly larger than the other, even if a topic only appears in one collection. For example, the two "junk" topics shown on the left are examples of this situation. In addition, if two topics are common to both data collections but one is popular among a lot of short documents (e.g., Twitter messages) and the other is prevalent in a relative small number of long documents (e.g., news articles), some sort of normalization schemes is clearly needed. Although there exist some sophisticated ranking heuristics [11], we argue that our model can handle these issues in a more principled way by modeling common topics explicitly. For instance the α

values for common topics can shed light on how popular these topics are, either in one of the data collections or in all of them.

Since similar ranking heuristics do not work well for `LDA` to provide local topics for Twitter and news, we only report the local topics found by our method, shown in the lower part of Table 8.1. On the left-hand side top ranked local topics on news are presented while on the right hand side top local topics found on Twitter are shown. Interesting observations can be made based upon these results. First, news articles tend to have more "formal" topics, such as politics, education and economy, whereas a large fraction of the Twitter stream consists of personal chat and opinions. Therefore, besides the common topics (e.g., Table 8.1) in both news and Twitter, local topics for Twitter seem less understandable and coherent. Indeed, throughout several experiments conducted on May's data and on other months as well, we observe that most of the local topics on Twitter are not very interesting. On the other hand, based on our experiments, some local topics (e.g "Crime") are on news but seldom picked up on Twitter. Many different kinds of criminal incidents are reported on a variety of news sources but not many of them really trigger interest on Twitter. Note that we understand that these results are preliminary and more thorough experiments are required. Nevertheless, our method provides a tool to investigate these interesting phenomena which are difficult or were impossible to be examined before [226].

8.4.3 Case Study on A Common Topic

Besides finding common and local topics on news and Twitter, our model also provides information about the temporal dynamics of these topics. Here we take a topic related to "Kentucky Derby" as an example to show the usefulness of our method. The Kentucky Derby[6] is the premier annual American horse race and has a significant international following. In 2010 it took place on May 1st. We try to identify the topics related to this

[6]http://en.wikipedia.org/wiki/Kentucky_Derby

event from the results obtained by our model. Remember that topics are only distributions over words. In order to find potential topics, we check the ranking positions of a list of terms which are known to be related to the event (e.g., "horse", "race", "kentucky", "derby"). If these terms are ranked highly in a particular topic, we consider that topic to be about the Kentucky Derby. We list the top 5 ranked terms of the topic we found by this simple heuristic just described: "derby", "race", "borel", "kentucky" and "horse". The topic we matched is a **common** topic and therefore it has the same distribution over words for both News and Twitter, meaning that once an article in News or a message in Twitter refers to this topic, the same word distribution is used to generate words, which is guaranteed by the model. However, the difference between News and Twitter on how this topic would be selected in a document is controlled by a stream-specific prior $\sim \text{Dir}(\alpha_t^{(c)})$ and further governed by a stream-dependent temporal gamma function.

In order to show the time series of the topic on news and Twitter, we transform the counts into a valid distribution by calculating a $p(t|z) = \frac{p(z|t)p(t)}{\sum_{t'} p(z|t')p(t')}$ where t is a time epoch. Then, $p(z|t)$ is estimated by the number of tokens assigned to topic z in time epoch t divided by the total number of tokens in time epoch t and $p(t)$ is estimated by the total number of tokens in time t divided by the overall number of tokens across all time epochs. Basically, the probability $p(t|z)$ tells us how likely the topic would appear in time epoch t. The results are shown in Figure 8.4 where the X-axis is the hours in May, 2010. Y-axis is $p(t|z)$. We first show the topic on the whole timeline (720 hours in May) on the top and show the first 120 hours at the bottom of the figure. The first observation is that the topic has two major peaks on both news and Twitter, shown in the upper part. This may reflect that "Kentucky Derby" is indeed a popular sports event. From the first 120-hour view of the topic, it is interesting to see that the topic first exhibited a peak on News and exhibited another peak on Twitter several hours later. This is a concrete example demonstrating the potential usage of our model to analyze common topics on

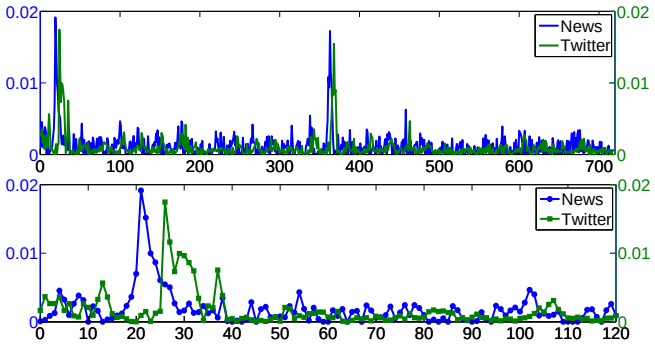

Figure 8.4: Temporal dynamics of "Kentucky Derby" on News and Twitter.

multiple text streams in the timeline. A similar kind of analysis is conducted in [226] using sophisticated heuristics to find common topics and to view the timelines of topics.

8.4.4 Case Study on Hashtags

Hashtags, a type of community convention[7] which starts with a "#" sign, have been heavily used as annotations to represent events and topics on Twitter. We select several hashtags that can act as indicators for certain events where each hashtag is clearly associated to some events in May, 2010. More specifically, we choose #mothersday for "Mothers Day", #memorialday for "Memorial Day", #bp for "Oil Spill",#kentuckyderby for "Kentucky Derby", #gaga for "Lady Gaga" and #justinbieber for "Justin Bieber". We wish to see whether these events can be discovered by different models and how well these topics can be presented. We believe these hashtags represent a large range of social events and therefore are representative. In order to make a fair comparison, we transform the volume of these hashtags over time into distributions by using a technique similar to those introduced above. The first question we want to ask is whether the models can identify topics that reflect the events behind these hashtags. We map hashtags onto the topics obtained by the models and top ranked terms in these topics are examined to see whether these terms

[7]http://twitter.pbworks.com/Hashtags

Table 8.2: Hashtag-to-Topic Mappings

Hashtag	**Top Terms of Mapped Topic**
[a] Hashtag Mapping for `LDA` model	
#mothersday	family home life children mother son friends
#memorialday	event june call center community club park
#bp	oil gulf spill coast mexico gas drilling
#kentuckyderby	race car track kentucky win top cars
#gaga & #justinbieber	justin lady super try bieber ider rio gaga jonas
[b] Hashtag Mapping for `Temporal Collection` model	
#mothersday	family children day home life church mother
#memorialday	memorial event day june community center
#bp	oil gulf spill coast drilling mexico water louisiana
#kentuckyderby	derby race borel kentucky horse super
#gaga & #justinbieber	bieber music video song gaga album lady

[c] KL Divergence between Hashtags and Matched Topics

Hashtag	`LDA` vs. `Temporal Collection`
#mothersday	1.1911 / 0.7714
#memorialday	1.4331 / 0.9365
#bp	0.3958 / 0.1577
#kentuckyderby	1.9924 / 0.8183
#gaga & #justinbieber	2.2391 / 1.1754

have any relationships with the underlying events. To map the hashtags, we calculate the following probability $p(z|w) = \frac{p(w|z)p(z)}{\sum_{z'} p(w|z')p(z')}$ where $p(w|z)$ is exactly $\phi_{z,w}$, provided by the trained models and $p(z)$ can be easily estimated by the counts. Intuitively, this probability tells us how likely a topic is to be selected, given the term. For the `Temporal Collection` model, all topics (including common topics and local topics) are treated as candidates to be matched. We map hashtags to topics for both `LDA` and our model, shown in the upper part of Table 8.2. Both models map #gaga and #justinbieber together onto a single topic, indicating that topics obtained by these models do not strictly correspond to real world events. Although some top ranked terms are similar for both models, the results from the `Temporal Collection` model are arguably better, in terms of *interpretation* of these terms. For instance, the `Temporal Collection` model explicitly ranks terms "memorial" and "day" highly in the list, implying this topic has much closer relationship with "Memorial Day", while `LDA` only has terms with broader connections with this kind of event. Similarly, the `Temporal Collection` model ranks more specific terms highly for

'Kentucky Derby" (e.g., "borel", "horse", "pletcher") while the topic obtained by LDA is essentially related to many races including "car races" and "horse races".

We can also compare the time-series of topics and hashtags to determine whether they are similar. The assumption is that, if they behave similarly on the timeline, the topics might be good choices for *explaining* the underlying events. Note that we are **not** seeking the **exact** match here since the topics have many more terms rather than a single hashtag and it may explain multiple events. Again, we transform the volumes into probabilities. We plot the time series of selected hashtags and the time series of selected topics in the same plots, shown in Figure 8.5 where X-axis is the hour number and Y-axis is the probability. For each hashtag we compare its time series, obtained using LDA, with those obtained from our model. Although top ranked terms may look similar, the time series of these topics behave significantly differently. For LDA, because of the fixed Dirichlet hyper-parameter α over time, the models may give inappropriate "pseudo counts" for certain topics in the timeline. Indeed, one property of Dirichlet distribution can shed some lights on the observation: $\mathbb{E}[\theta_k] = \frac{\alpha_k}{\sum_{k'} \alpha_{k'}}$ where the expected value of θ_k, the proportion of topic k represented in a document, is the ratio of the Dirichlet parameter α_k over the sum of all α values. Since α values are fixed over time, this expected value will also be constant over time, leading to the fact that the topic assignments fluctuate around a certain value, though with variance, which is exactly shown in our experiments. This drawback of LDA may lead to difficulty in identifying the peaks of these topics. On the other hand, in our model, since the hyper-parameters α are controlled by the temporal gamma functions, the rise and fall of these values may give good hints for the model to assign topics to words, yielding better modeling temporal dynamics. Also, from the results, our models can better match the peaks of hashtags, indicating that the method can better reflect real events.

We can further compare these time series quantitatively. Since these time series are valid distributions over time, KL divergence is employed to measure their "distance" as

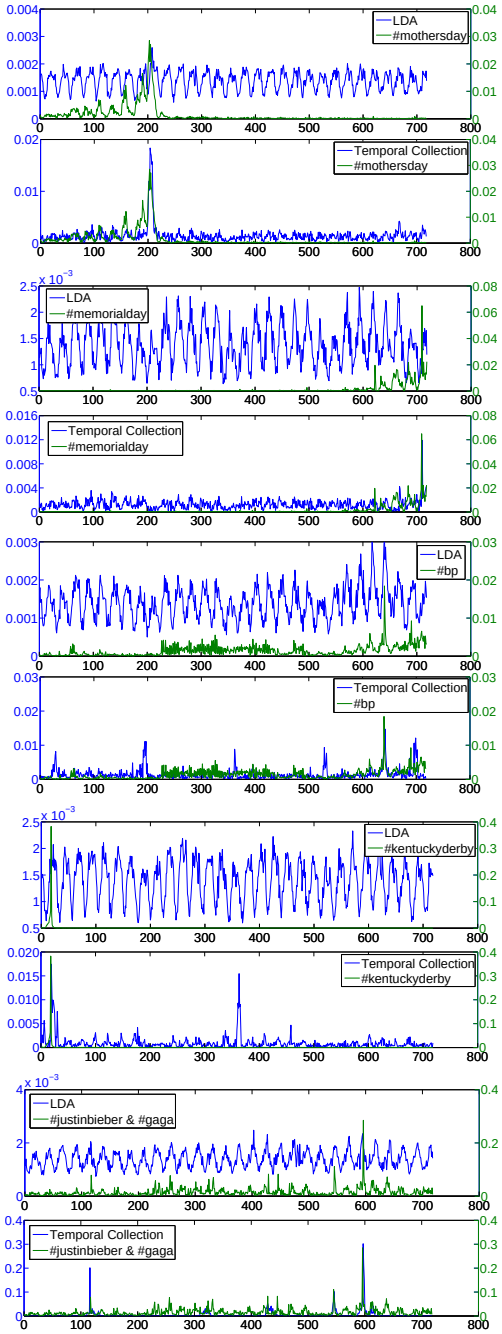

Figure 8.5: The distributions $p(t|z)$ of mapped topics in May.

Table 8.3: Evaluation on Retrieval Performance

Method	**MAP**
TF-IDF	0.673
TF-IDF + Plain `LDA`	0.685
TF-IDF + `Collection`	0.703
TF-IDF + `Temporal Collection`	0.732

follows: $\sum_t p(t|w_1) \log \frac{p(t|w_1)}{p(t|w_2)}$. KL divergence is non-negative and the smaller the value is, the more similar two distributions are. We calculate the KL divergence between hashtag time series and topic time series for both `LDA` and our model. The results are shown in the bottom part of Table 8.2. From the results it is obvious that the time series of topics obtained by the `Temporal Collection` model better match the corresponding hashtag time series, yielding lower KL divergence scores. This also validates the visual evidence from Figure 8.5.

8.4.5 Performance on Retrieval

As a further demonstration of the utility and effectiveness of our model, we apply it in a toy application that uses it as part of an information-retrieval relevance measure. For a query q and document d, the idea is to use the probability $p(q|d)$ that q was generated by d's generating model as a measure of the relevance of d to q. In a scheme similar to that used in [204] we use a relevance measure $S(d|q)$ that is a linear combination of $p(q|d)$ and a simple TF-IDF-based cosine-similarity score $\tau(d, q)$. That is

$$S(d|q) \quad = \quad \lambda\, \tau(d, q) \;+\; (1 - \lambda)\, p(q|d). \tag{8.13}$$

For our experiment we select the top 20 queries from `GoogleInsights` in the time period of May 2010, corresponding to our datasets. To select retrieval candidates we compute $\tau(d, q)$ for all tweet-query pairs (d, q) and use these scores to rank the tweets for all queries. We

then select the top 50 tweets from that ranking for each query. These tweets and queries are then submitted to Amazon Mechanical Turk for manual relevance judgements, which we use as ground truth. These judgements are assigned using a three-level scale consisting of "relevant", "neutral" and "non-relevant." For each pair (d, q) three judges are assigned to assess the relevance and only the pairs on which at least two workers agree are kept, leaving a total of 922 tweets.

Mean Average Precision (MAP) for the top 20 positions is used for retrieval-accuracy characterization. These top-20 MAP scores are computed for each of four combinations of the TF-IDF measure $\tau(d, q)$ and a topic model. For each combination the parameter λ of (8.13) is varried over the range $[0, 1]$ and the optimal (highest MAP) value is determined. The corresponding MAP values are shown in Table 8.3. From these results we see that the choice of topic model used affects retrieval accuracy, with the highest retrieval accuracy being associated with the combination of TF-IDF and `Temporal Collection` scores.

8.5 Summary

Modeling the temporal dynamics of topics is still a challenge, especially on multiple data collections. In this chapter we propose a model for use in automatically analyzing multiple correlated text streams with their temporal behavior in a principled way. Our method bridges the recent advances in topic-modeling and information cascading in social media. We extend topic models by allowing each text stream to have local topics and shared topics, overcoming several theoretical problems of previously proposed models for similar problems. For temporal modeling we associate each topic with a time-dependent function that characterizes its popularity over time. By combining the two models, we can effectively model the temporal dynamics of multiple correlated text streams in a unified framework. Compared to related work our method is easy to implement and can potentially scale to

large datasets. Additionally our method provides a new tool for browsing and mining a variety of types of social media simultaneously. For future work it will be interesting to utilize Bayesian non-parametric techniques to automatically learn the number of topics from the dataset. This is especially valuable for our model where the number of common topics and local topics must be manually assigned in current settings. In addition in order to better reflect real events, topics can be linked with named entities such that each topic is forced to contain a certain number of entities. It is also interesting to see hierarchical modeling of topics with temporal dynamics, which permits users to "zoom in" and "zoom out" on large topics (e.g. "oil spill") and track their evolution over time.

8.6 Bibliographic Notes

Mining common topics and their temporal dynamics from multiple text streams can be loosely decomposed into the two independent tasks of (1) recovering topics and (2) characterizing their temporal dynamics. We review these two lines of related work.

Based on PLSA, Wang et al. [198] introduced an observed-time-stamp variable into the generative model to incorporate temporal dynamics. In addition several heuristics were applied to smooth topics in consecutive time periods. Later, Wang et al. [199] followed a similar idea and used an artificial time-synchronization optimization process in their model to re-organize the time stamps of all documents so that documents with the same time stamp would share similar topics. We argue that the constraint imposed by this synchronization is unrealistic. Note that these two s do not differentiate between common topics and topics that only occur in a single text stream. Moreover, since both models are based on PLSA, they have the tendency to overfit the data. Furthermore, both models are not well-defined generative models [28] and no assumptions on how topic distributions and per-document topic-proportion distributions change over time were made in these

models. In a recent paper, Zhang et al. [224], in addressing the same problem, proposed a non-parametric model in which a Markovian assumption is made regarding the temporal dynamics of document-topic distributions. As mentioned in the previous section, however, according to recent results on information propagation and temporal variations [130, 213], this assumption may not be appropriate for social media.

Independent of temporal factors, two basic approaches to topic discovery from correlated text streams exist in the topic modeling literature. Zhai et al. [222] proposed two variants of the same idea to tackle the problem of modeling multiple text streams. One variant assumes that each document in a text stream is generated by a background language model and a set of topics. Both the background language model and topics are multinomial distributions over words shared across multiple text streams. Since they are shared across all streams, common topics are difficult to identify. The second variant also assumes that each document in a text stream is generated by a background language model and a set of topics. Once a term is chosen to be generated by topics, a topic index is first selected followed by a second-level decision regarding whether the word is generated by a common or a local topic. The model can then explicitly handle common and local topics among multiple streams. Common and local topics are aligned under the same set of indices however, forcing the total number of topics to be the same for all streams. In addition the background language is the same across all text streams, which is too inflexible for the joint modeling of disparate sources such as Twitter and Yahoo! News. Also, per-document topic-proportion parameters must be manually tuned in experiments, which is impractical for real applications. The first variant inspired models introduced in [198, 199] and the second variant was extended to a fully Bayesian formulation by Paul et al. [154, 155], in which the topic proportion parameters were automatically estimated from the inference algorithm but local topics among different text streams were forcibly put under the same set of indices. It is therefore possible that unrelated topics will be

brought together under the same topic index due to this constraint.

We briefly review some of the recent extensive work on modeling temporal dynamics in topic models. Early work on incorporating temporal evolution usually made a Markovian assumption by using either a state-space model (e.g., [26, 196]) or a linear model (e.g., [161]). Besides the Markovian assumption, Wang et al. [201] introduced a beta distribution over timestamps using a non-Markovian topic model. Nallapati et al. [148] and Iwata et al. [103] focused on the problem of modeling topics spread on a timeline with multiple resolutions, namely how topics are organized in a hierarchy and how they evolve over time. Ahmed and Xing [6] proposed a non-parametric model to address the birth and death of topics over a timeline using a Markovian assumption. The datasets used in these papers are several orders of magnitude smaller than the one we used in this chapter.

Chapter 9

Topic Modeling: Temporal Modeling by Tracking Trends

Text corpora with documents covering a long time-span are natural and ubiquitous in many application fields, and include such data as research papers and newspaper articles. Mining from these collections, discovering and understanding underlying topics and ideas, continues to be an important task. In addition to traditional text collections, many types of content in social media make applying machine learning techniques to these new data sources more challenging, such as forums, question answering communities and blog entries. People not only would like to know what kind of topics can be found from these data sources but also wish to understand the temporal dynamics of these topics, and hopefully predict certain properties of terms or documents in the future.

Topic models like [28]), as a class of newly developed machine learning tools, have been studied extensively in recent years. From the seminal work done by Blei et al. [28], a large body of literature about topic models has been established. Multiple disciplines of computer science, ranging from information retrieval (e.g., [204]), computer vision (e.g., [200]) to collaborative filtering (e.g., [2]) have applied topic models to their problems. For

text modeling, topic models are applied to find latent topics from text collections, which is particularly useful for temporal text corpora where discovered latent topics can help researchers visualize and understand the thematic evolution of the corpora over time. This has led to the recent development of incorporating temporal dynamics into topic models (e.g., [143, 26, 201, 142, 148, 198, 196, 140, 199, 224, 6, 103, 111]). These models enable us to browse and explore datasets with temporal changes in a convenient way and open future directions for utilizing these models in a more comprehensive fashion. One drawback of these existing models is that most of them are general purpose models with which no real tasks are explicitly associated. Therefore, it might be difficult to employ these models in real-world applications, such as the problems of tracking trends and predicting popularity of keywords. As a result of the lack of a particular task, there is also no consensus on how these models should be evaluated and compared. Although perplexity is widely used in these papers, as pointed out in [31], this measure may not have correlations with the quality (e.g., coherence) of topics discovered. Furthermore, no empirical or theoretical work has been done as far as we know to show the the correlations between the low perplexity values and high performance in third-party tasks such classification, regression and clustering. In this chpater, we argue that temporal topic models should be evaluated on specific real-world tasks and propose such a task to compare how they can contribute to applications. Some recent extensions of topic models (e.g., [27, 126, 229, 164]) have tried to incorporate side information, such as document-level labels and word-level features (e.g., [158]) into models in order to perform classification and regression tasks. A basic conclusion made from these attempts is that these special-purposed models, aiming to optimize particular tasks, perform better than general-purpose models, on the tasks they evaluated. We share a similar spirit in this chapter, showing that temporal topic models for special tasks perform better than general-purpose models.

In this chapter, we introduce a real-world task — tracking trends of terms — to which

temporal topic models can be applied. Rather than building a general-purpose model, we propose a new type of topic model incorporating the volume of terms into the temporal dynamics of topics and directly optimize for the task. Unlike existing models in which trends are either latent variables or not considered at all and thus are difficult to apply in practice, we combine state-space models with term volumes in a supervised learning fashion which enables us to effectively predict volumes in the future, even without new documents. In addition, it is straightforward to obtain the volumes of latent topics as a by-product of our model, demonstrating the superiority of utilizing temporal topic models over traditional time-series tools (e.g., autoregressive models) to tackle this kind of problem. The proposed model can be further extended with arbitrary word-level features which are evolving over time. We present the results of applying the model to two datasets with long time periods and show its effectiveness over non-trivial baselines. Our contributions are threefold:

- Introduce a task — volume tracking — that can be used as a standard evaluation method for temporal topic models
- Propose a temporal topic model that directly optimizes the task introduced
- Demonstrate the effectiveness of the model as compared to state-of-the-art algorithms by experimenting on two real-world datasets

We organize the chapter as follows. In Section 9.6, we review some related developments of topic models and existing evaluation methods for temporal topic models. In Section 9.1, we introduce the task of volume tracking, as a case of trend monitoring, and propose our model . In Section 9.2, we show how to utilize variational inference with Kalman Filter to estimate hidden parameters of the model. In Section 9.3, we discuss some other models that can be used in the volume tracking task. In Section 9.4, we demonstrate the experimental results on two datasets and conclude the chapter in Section 9.5.

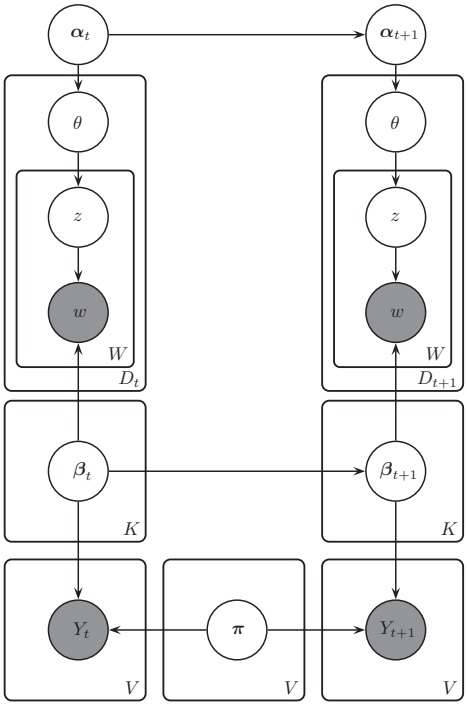

Figure 9.1: A graphical representation of the model.

9.1 Tracking trends by incorporating volumes

In this section, we will introduce the task of volume prediction as a case of trend tracking. One reason that temporal topic models are favored is perhaps that these models can be potentially used as a tool to analyze trends and changes of keywords over time. However, these tasks are never evaluated directly or seriously in current literature.

The task of predicting the volume of terms is to predict the numeric volume of one or a set of keywords, given the historical data of these keywords in the past. This is a natural extension of tracking and monitoring keywords over time. Indeed, some commercial products provide such tools to allow users to browse and understand the rise and fall of keywords, such as `Google Trends`. One drawback of existing tools is that people usually only have a limited view of certain topics in which they are interested before they

fully understand these topics. For instance, for the event of "World Cup", the phrase "World Cup" is certainly of interest. However, there are many more related terms to be explored, such as "FIFA", "South Africa" and "Ronaldo". Sometimes, users have these related terms in mind but usually they are unable to prepare them in advance. It would be great if users could track the trends (volume) of a topic as a whole and discover all those related terms at the same time. Moreover, the volume of terms in the same topic are correlated, which may help the model to find better topics. Overall, we would like to achieve three goals in tracking trends:

- Track and predict the volume of individual terms
- Obtain latent topics so that related terms can be grouped together
- Model the evolution of latent topics

The second goal will happen automatically through the modeling of topic models. The last goal can be achieved by temporal topic models, through either one of the assumptions mentioned in Section 9.6. The first goal is the center of this work. We believe that our work would help to track the volume of topics as a whole if the first goal can be achieved. Note, in terms of "prediction", we indicate the ability to estimate the volume of individual terms in the future where no documents are realized. Two design issues need to be tackled when introducing term volumes into the model. First, they are word-level variables (if we treat features as random variables). Second, we need to predict values of these variables without documents. These two issues prevent these variables from being placed in the document plates, in terms of graphical modeling. This decision distinguishes our model from previous models (e.g., [27, 126, 229, 164]) where response variables are placed in document plates. Recently, Petterson et al. [158] demonstrate a technique to embed word-level features into topic models. Although our work shares similar ideas to theirs, their model is not a generative model for word features but only for words in the

documents. In addition, their work is not to predict these word-level features. Since their work is for a static text corpus, it cannot be easily utilized to model temporal data. Therefore, we do not include this model in our experiments for comparison. Our model is a fully generative model for both word instantiations in documents and word-level features. Before we further go to the formal description of our model, we discuss some intuitions behind the model. In standard topic models, each word v is associated with many latent topics $\boldsymbol{\beta}_{1:K}$. Each topic β_k is a distribution over all terms in the vocabuary V. Intuitively, the more a term appears in many topics, the more likely the term will have a high volume, such as some stop words and functional words. On the other hand, many terms only appear in a handful of topics and therefore these topics determine the volume of the term. If we think of $\boldsymbol{\beta}$ as another representation of terms, we would like to associate these latent variables with the term volumes. Following this intuition, we treat the volume of term v at time-stamp t, denoted as $Y_v^{(t)}$, as a function of latent topics $\boldsymbol{\beta}$. The simplest form of such functions is a linear function:

$$Y_v^{(t)} = \sum_{k=0}^{K} \pi_{(v,k)} \beta_{(k,v)}^{(t)} + \epsilon_v \tag{9.1}$$

where $\boldsymbol{\pi_v}$ is a vector of coefficients, $\beta_{(k,v)}^{(t)}$ is the probability that the term is "generated" from topic k at time stamp t, and ϵ_v is a per-term "error". In other words, the volume of a term v depends on its prevalence in all topics at that time point. If ϵ_v follows a normal distribution, namely $\epsilon_v \sim N(0, \sigma_v^2)$, we can express the generation process of $Y_V^{(t)}$ in terms of a Normal distribution as follows:

$$Y_v^{(t)} \mid \pi_{(v)}, \beta_{(*,v)}^{(t)} \sim \mathcal{N}\Big(\pi_v^T \beta_{(*,v)}^{(t)}, \sigma_v^2\Big)$$

Here, $Y_v^{(t)}$ is treated as a real valued variable. In our experiments, we use the raw counts

of term v at time epoch t as $Y_v^{(t)}$.

In order to obtain Y_v at different time epochs, we need to have $\boldsymbol{\beta}$ for different time points. We mention two basic categories of approaches in Section 9.6 and here we adapt the first category, having a "Markovian assumption" on the evolution of topics over time. More specifically, topics $\boldsymbol{\beta}$ evolve according to a state-space model and the documents with their words are "generated" by the corresponding topics in the same time epoch. Embedding these intuitions into the model, the generative process of the model is as follows:

1. For each topic k in K:
   Draw topics $\beta_k^{(t)} \mid \beta_k^{(t-1)} \sim \mathcal{N}\Big(\beta_k^{(t-1)}, \delta^2 I\Big)$.
2. For each term v in V:
   Draw term volume $Y_v^{(t)} \sim \mathcal{N}\Big(\pi_v^T \beta_{(*,v)}^{(t)}, \sigma^2\Big)$.
3. For each document d in time epoch t:
   (a) Draw $\theta_d \sim \text{Dir}(\boldsymbol{\alpha})$
   (b) For each word n:
      i. Draw $z_{(d,n)} \sim \text{Multi}(\theta)$.
      ii. Draw $w_{(d,n)} \sim \text{Multi}\Big(f(\beta_z^{(t)})\Big)$

where function f maps the multinomial natural parameters to mean parameters. The graphical representation of the model is shown in Figure 9.1 where only two epocs are shown. Note, the model can be easily extended in multiple ways. For instance, we can also allow the hyper-parameters of topic proportions $\boldsymbol{\alpha}$ to evolve over time, according to a different state-space model, as already mentioned in [26]. In addition, the simple state-space model can be replaced by a Brownian motion model [196], allowing arbitrary granularity of time-series. We will explore these extensions in future work.

9.2 Variational Inference with Kalman Filtering

The central problem in topic modeling is posterior inference, i.e., determining the distribution of the latent topic structure conditioned on the observed documents. In our case, the latent structures comprise the per-document topic proportions θ_d, per-word topic assignments $z_{(d,n)}$, the K sequences of topic distributions $\beta_k^{(t)}$ and per-term coefficient vector $\boldsymbol{\pi}_v$ for characterizing term volumes. Similar to many topic models, the true posterior is intractable [26, 196], meaning that we must appeal to an approximation.

Several approximate inference approaches have been developed for topic models. The most widely used are variational inference (e.g., [28, 26, 196]) and collapsed Gibbs sampling (e.g., [79, 201]). As noted previously by others [26, 196], collapsed Gibbs sampling is not an option in the sequential setting because the distribution of words for each topic is not conjugate to the word probabilities. Therefore, we employ variational inference for the model.

The main idea behind variational inference is to posit a simple family of distributions over the latent variables, namely variational distributions, and to find the member of that family which is closest in Kullback-Leibler divergence to the true posterior. Variational inference has been successfully adopted in temporal topic models (e.g., [26, 148, 196]).

For the model descried above, we adapt variational Kalman filtering [26] to the sequential modeling setting. We employ the following variational distribution:

$$q(\boldsymbol{\beta}_{1:T}, \boldsymbol{\theta}, \mathbf{Z} | \hat{\boldsymbol{\beta}}_{1:T}, \boldsymbol{\lambda}, \boldsymbol{\Phi}) = \prod_{k=1}^{K} q(\beta_k^1, \cdots, \beta_k^T | \hat{\beta}_k^1, \cdots, \hat{\beta}_k^T) \times \prod_{t=1}^{T} \Big(\prod_{d=1}^{D_t} q(\theta_d | \lambda_d) \prod_{n=1}^{N_d} q(z_{(d,n)} | \phi_{(d,n)}) \Big)$$

The variational parameters are a Dirichlet λ_d for the per-document topic proportions, multinomials ϕ for each word's topic assignment, and $\hat{\beta}$ variables, which are "observations" to a Variational Kalman Filter. The central idea of the variational Kalman filter is that variational parameters are treated as "observations" in a common Kalman filter setting,

while true parameters, here $\boldsymbol{\beta}^{(t)}$, are treated as latent states of the model. By utilizing a Kalman filter, we can effectively estimate these "latent states" through "observations".

More specifically, our state space model is:

$$\begin{aligned} \beta_k^{(t)} \,|\, \beta_k^{(t-1)} &\sim \mathcal{N}\Big(\beta_k^{(t-1)}, \delta^2 I\Big) \\ \hat{\beta}_k^{(t)} \,|\, \beta_k^{(t)} &\sim \mathcal{N}\Big(\beta_k^{t}, \hat{\delta}_t^2 I\Big) \end{aligned} \tag{9.2}$$

The variational parameters are $\hat{\beta}_k^{(t)}$ and $\hat{\delta}_t$. The key problem of Kalman filter is to derive the mean and variance for forward and backward equations, which can be used to calculate the lower bound in variational inference. Using the standard Kalman filter calculation, the forward mean and variance of the variational posterior are given by:

$$\begin{aligned} m_k^t = \mathbb{E}[\beta_k^t | \hat{\boldsymbol{\beta}}_k^{1:t}] &= \left(\frac{\hat{\delta}^2}{V_k^{t-1} + \delta^2 + \hat{\delta}^2}\right) m_k^{t-1} + \left(1 - \frac{\hat{\delta}^2}{V_k^{t-1} + \delta^2 + \hat{\delta}^2}\right) \hat{\beta}_k^t \\ V_k^t = \mathbb{E}\Big[(\beta_k^t - m_k^t]) \,|\, \hat{\boldsymbol{\beta}}_k^{1:t}\Big] &= \left(\frac{\hat{\delta}^2}{V_k^{t-1} + \delta^2 + \hat{\delta}^2}\right) (V_k^{t-1} + \delta^2) \end{aligned} \tag{9.3}$$

with initial conditions specified by fixed m^0 and V^0. The backward recursion then calculates the marginal mean and variance of β_k^t given $\hat{\boldsymbol{\beta}}_k^{1:T}$ as:

$$\begin{aligned} \widetilde{m}_k^{t-1} = \mathbb{E}[\beta_k^{t-1} | \hat{\boldsymbol{\beta}}_k^{1:T}] &= \left(\frac{\delta^2}{V_k^{t-1} + \delta^2}\right) m_k^{t-1} + \left(1 - \frac{\delta^2}{V_k^{t-1}] + \delta^2}\right) \widetilde{m}_k^t \\ \widetilde{V}_k^{t-1} = \mathbb{E}\Big[(\beta_k^{t-1} - \widetilde{m}_k^{t-1}]) \,|\, \hat{\boldsymbol{\beta}}_k^{1:T}\Big] &= V_k^{t-1} + \left(\frac{V_k^{t-1}}{V_k^{t-1} + \delta^2}\right)^2 \Big(\widetilde{V}_k^t - (V_k^{t-1} + \delta^2)\Big) \end{aligned} \tag{9.4}$$

with initial conditions $\widetilde{m}^T = m^T$ and $\widetilde{V}^T = V^T$.

With these forward and backward equations in hand, we turn to calculate the following lower bound (assuming $\boldsymbol{\Omega} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\pi}, \sigma^2\}$) with the help of variational distributions

introduced in Equation 9.3:

$$\log P(\mathbf{W}, \mathbf{Y}|\mathbf{\Omega}) \geq \mathbb{E}_q[\log p(\boldsymbol{\beta})] + \mathbb{E}_q[\log p(\mathbf{W}, \mathbf{Z}, \boldsymbol{\theta}|\boldsymbol{\beta}, \boldsymbol{\alpha})] + \mathbb{E}_q[\log p(\mathbf{Y}|\boldsymbol{\pi}, \boldsymbol{\beta}, \sigma^2)] + H(q)$$
$$= \mathbb{E}_q[\log p(\boldsymbol{\beta})] + \mathbb{E}_q[\log p(\mathbf{W}|\mathbf{Z}, \boldsymbol{\beta})] + \mathbb{E}_q[\log p(\mathbf{Z}|\boldsymbol{\theta})] + \mathbb{E}_q[\log p(\boldsymbol{\theta}|\boldsymbol{\alpha})] + \mathbb{E}_q[\log p(\mathbf{Y}|\boldsymbol{\pi}, \boldsymbol{\beta}, \sigma^2)] + H(q) \quad (9.5)$$

where term $H(q)$ is the entropy. To tighten the above bound on the likelihood of the observations given by Jensen's inequality is equivalent to minimize KL-divergence. In the above bound, the term $\mathbb{E}_q[\log p(\mathbf{W}, \mathbf{Z}, \boldsymbol{\theta}|\boldsymbol{\beta}, \boldsymbol{\alpha})]$ is standard for topic models, when logistic-normal distribution is applied to represent topics (e.g., [26, 196]. The term $\mathbb{E}_q[\log p(\boldsymbol{\beta})]$ is standard for temporal topic models, which utilize the Kalman filter as a sequantial modeling tool. The term $\mathbb{E}_q[\log p(\mathbf{Y}|\boldsymbol{\pi}, \boldsymbol{\beta}, \sigma^2)]$ can be calculated similarly to the document-level response variables, introduced in [27]. We will discuss these expectations in detail.

For the first term of the last line in Equation 9.5, we utilize the forward and backward equations introduced in Equation 9.4 and follow the similar steps in [26]:

$$\mathbb{E}_q[\log p(\boldsymbol{\beta})] = -\frac{VKT}{2}(\log \delta^2 + \log 2\pi) - \frac{1}{2\delta^2}\sum_{t=1}^{T}\sum_{k=1}^{K}\left[\left(\widetilde{m}_k^t - \widetilde{m}_k^{t-1}\right)^2\right] - \frac{1}{\delta^2}\sum_{t=1}^{T}\sum_{k=1}^{K}\mathrm{Tr}\left(\widetilde{V}_k^t\right)$$
$$+ \frac{1}{2\delta^2}\sum_{k=1}^{K}\mathrm{Tr}\left(\widetilde{V}_k^T\right) - \frac{1}{2\delta^2}\sum_{k=1}^{K}\mathrm{Tr}\left(\widetilde{V}_k^0\right)$$

For the second term in the same line, we have:

$$\mathbb{E}_q[\log p(\mathbf{W}|\mathbf{Z}, \boldsymbol{\beta})] = \sum_{t=1}^{T}\sum_{d=1}^{D_t}\sum_{n=1}^{N_d}\left(\sum_{k=1}^{K}\phi_{(n,k)}\widetilde{m}_{(k,w)}^t - \sum_{k=1}^{K}\phi_{(n,k)}\mathbb{E}_q\Big[\log\sum_{w'}\exp(\beta_{(k,w')})\Big]\right)$$

where the second line demonstrates the essential problem of non-conjugacy of using the logistic-normal distribution for topics. In order to calculate $\mathbb{E}_q\Big[\log\sum_{w'}\exp(\beta_{(k,w')})\Big]$, we further obtain a lower bound by introducing another variational parameter ζ_t and upper

bound the negative log normalizer with a Taylor expansion as follows:

$$\mathbb{E}_q\Big[\log\sum_{w'}\exp(\beta_{(k,w')})\Big]\leq\zeta_t^{-1}\Big(\sum_{w'}\mathbb{E}_q[\exp(\beta_{(k,w')})]\Big)-1+\log(\zeta_t)$$

where the expectation $\mathbb{E}_q[\exp(\beta_{(k,w')})]$ is the mean of a log normal distribution with the mean and variance obtained from the variational parameters, essentially Kalman Filters, in our case. For the third term of the last line in Equation 9.5, we have:

$$\mathbb{E}_q[\log p(\mathbf{Z}|\boldsymbol{\theta})]=\sum_{t=1}^{T}\sum_{d=1}^{D_t}\sum_{n=1}^{N_d}\sum_{k=1}^{K}\phi_{(n,k)}\Big[\Psi(\lambda_{(d,k)})-\Psi\Big(\sum_{k'=1}^{K}\lambda_{(d,k')}\Big)\Big]$$

and for the fourth term, we have:

$$\mathbb{E}_q[\log p(\boldsymbol{\theta}|\boldsymbol{\alpha})]=\sum_{t=1}^{T}\sum_{d=1}^{D_t}\Bigg\{\Bigg(\sum_{k=1}^{K}(\alpha_k-1)\Big[\Psi(\lambda_{(d,k)})-\Psi\Big(\sum_{j=1}^{K}\lambda_{(d,j)}\Big)\Big]\Bigg)+\log\Gamma\Big(\sum_{k=1}^{K}\alpha_k\Big)-\sum_{k=1}^{K}\log\Gamma(\alpha_k)\Bigg\}$$

For the last term in the same line, we have:

$$\begin{aligned}&\mathbb{E}_q[\log p(Y_v^{(t)}|\pi_v,\beta_{(,v)}^{(t)},\sigma^2)]=-\frac{1}{2}\log 2\pi-\frac{1}{2}\log\sigma^2\\&-\frac{\Big(Y_v^{(t)}\Big)^2}{2}+\frac{1}{\sigma^2}\Bigg[Y_v^{(t)}\sum_{k=1}^{K}\pi_{(v,k)}\widetilde{m}_{(k,v)}^{t}-\frac{1}{2}\sum_{i=1}^{K}\sum_{j=1}^{K}\pi_{(v,i)}\Big(\widetilde{m}_{(i,v)}^{t}\widetilde{m}_{(j,v)}^{t}\Big)\pi_{(v,j)}\Bigg]\end{aligned}$$

For the entropy term $H(q)$, we have:

$$\begin{aligned}&-H(q)=\mathbb{E}_q[\log q(\boldsymbol{\beta}|\hat{\boldsymbol{\beta}})]+\mathbb{E}_q[\log q(\boldsymbol{\theta}|\boldsymbol{\lambda})]+\mathbb{E}_q[\log q(\mathbf{Z}|\boldsymbol{\Phi})]\\&=\sum_{t=1}^{T}\sum_{k=1}^{K}\Big(\frac{T}{2}\log 2\pi\Big)+\frac{1}{2}\sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{v=1}^{V}\log\widetilde{V}_{(k,v)}^{t}+\sum_{t=1}^{T}\sum_{d=1}^{D_t}\Bigg\{\Bigg(\sum_{k=1}^{K}(\lambda_{(d,k)}-1)\Big[\Psi(\lambda_{(d,k)})-\Psi\Big(\sum_{j=1}^{K}\lambda_{(d,j)}\Big)\Big]\Bigg)\\&+\log\Gamma\Big(\sum_{k=1}^{K}\lambda_{(d,k)}\Big)-\sum_{k=1}^{K}\log\Gamma(\lambda_{(d,k)})\Bigg\}+\sum_{t=1}^{T}\sum_{d=1}^{D_t}\sum_{n=1}^{N_d}\sum_{k=1}^{K}\phi_{(n,k)}\log\phi_{(n,k)}\end{aligned}$$

By using the expectations with respect to variational distributions, we can optimize the

Algorithm 2: Variational inference with Kalman filtering.

```
Initialize β̂ randomly.
while relative improvement in L > 0.00001 do
    E step:
    for t = 1 to T do
        for i = 1 to D do
            Update λ_d according to Equation 8
            Update φ_d according to Equation 9
        Update ζ_t according to Equation 10
    M step:
    for v = 1 to V do
        Update π_v according to Equation 12
        Update σ_v^2 according to Equation 13
    Update β̂ by using conjugate gradient descent
```

variational parameters as follows. For per-document parameters $\lambda_{(d,k)}$, per-word parameters ϕ_n and per time epoch parameters ζ_t, we have similar update equations as standard topic models:

$$\lambda_{(d,k)} = \alpha_k + \sum_{n=1}^{N_d} \phi_{(n,k)}$$

$$\phi_{(n,k)} \propto \exp\left(\Psi(\lambda_{(d,k)}) - \Psi\Big(\sum_{k'=1}^{K} \lambda_{(d,k')}\Big)\right) \times \exp\left(\widetilde{m}^t_{(k,w)} - \mathbb{E}_q\Big[\log \sum_{w'} \exp(\beta_{(k,w')})\Big]\right)$$

$$\zeta_t = \frac{1}{N_t} \sum_{d=1}^{D_t} \sum_{n=1}^{N_d} \left(\sum_{k=1}^{K} \phi_{(n,k)} \sum_{w} \exp\Big(\widetilde{m}^t_{(k,w)} + \widetilde{V}^t_{(k,w)}/2\Big)\right)$$

Since π_v is a vector of coefficients across all time epochs T, we gather the $\beta^*_{(*,v)}$ from all time epochs and form a $T \times K$ matrix X where each row is a vector of β values discussed before. We can obtain the following equation by using the notation of X:

$$\mathbb{E}_q[\mathbf{X}^T\mathbf{X}]\pi_v = \mathbb{E}_q[\mathbf{X}]^T\mathbf{Y}_v$$

and therefore, we have

$$\pi_v = \Big(\mathbb{E}_q[\mathbf{X}^T\mathbf{X}]\Big)^{-1}\mathbb{E}_q[\mathbf{X}]^T\mathbf{Y}_v$$

where the t^{th} row of $\mathbb{E}_q[\mathbf{X}]$ is just $\mathbb{E}_q[\beta^t_{(,v)}]$. Similar to linear regression but in the expected version, we can obtain the update equation for σ_v^2 as:

$$\sigma_v^2 = \frac{1}{T}\left(\mathbf{Y}_v^T\mathbf{Y}_v - 2\mathbf{Y}_v^T\mathbb{E}_q[\mathbf{X}]\pi_v + \pi_v^T\mathbb{E}_q[\mathbf{X}^T\mathbf{X}]\pi_v\right)$$

where π_v is the new estimate value.

The real computational hurdle is to calculate the updates of $\hat{\boldsymbol{\beta}}$. Gathering all terms in the lower bound involving $\boldsymbol{\beta}$ and differentiating them with respect to $\hat{\beta}^t_{(k,v)}$, we have:

$$\begin{aligned}
&-\frac{1}{\delta^2}\sum_{t=1}^{T}\Big(\widetilde{m}^t_{(k,v)} - \widetilde{m}^{t-1}_{(k,v)}\Big)\left(\frac{\partial\widetilde{m}^t_{(k,v)}}{\partial\hat{\beta}^t_{(k,v)}} - \frac{\partial\widetilde{m}^{t-1}_{(k,v)}}{\partial\hat{\beta}^t_{(k,v)}}\right)\\
&+\sum_{t=1}^{T}\left(N_{(t,v)}\phi_{(v,k)} - \sum_{v=1}^{V}N_{(t,v)}\phi_{(v,k)}\zeta_t^{-1}\exp\Big(m^t_{(k,v)}\right.\\
&\left.+\,V^t_{(k,v)}/2\Big)\right)\frac{\partial\widetilde{m}^t_{(k,v)}}{\partial\hat{\beta}^t_{(k,v)}} + \frac{1}{\sigma^2}\sum_{t=1}^{T}Y_v^t\pi_{(v,k)}\frac{\partial\widetilde{m}^t_{(k,v)}}{\partial\hat{\beta}^t_{(k,v)}}\\
&-\left[\frac{1}{2\sigma^2}\sum_{i=1}^{K}\sum_{j=1}^{K}\pi_{(v,i)}\Big(\widetilde{m}^t_{(i,v)}\widetilde{m}^t_{(j,v)}\Big)\pi_{(v,j)}\right]\frac{\partial\widetilde{m}^t_{(k,v)}}{\partial\hat{\beta}^t_{(k,v)}}
\end{aligned}$$

Unfortunately, no closed-form solution for $\hat{\beta}$ can be found. We adapt optimization techniques to obtain a local optimum of the $\hat{\beta}$ values. In our experiments, we utilize the conjugate gradient algorithm implemented in `GSL` library[1], which requires us to provide the gradients. The forward-backward equations for $\mathbb{E}_q$ can be used to derive a recurrence

[1] http://www.gnu.org/software/gsl/

for the gradients. The forward recurrence is:

$$\frac{\partial m_{(k,v)}^{t}}{\partial \hat{\beta}_{(k,v)}^{(s)}} = \left(\frac{\hat{\delta}^2}{V_k^{t-1} + \delta^2 + \hat{\delta}^2} \right) \frac{\partial m_k^{t-1}}{\partial \hat{\beta}_{(k,v)}^{s}} + \left(1 - \frac{\hat{\delta}^2}{V_k^{t-1} + \delta^2 + \hat{\delta}^2} \right) \mathbb{I}[s == t]$$

with the initial condition $\partial m_k^0 / \partial \hat{\beta_k}^s = 0$. The backward recurrence is then:

$$\frac{\partial \widetilde{m}_k^t}{\partial \hat{\beta_k}^s} = \left(\frac{\delta^2}{V_k^{t-1} + \delta^2} \right) \frac{\partial m_k^{t-1}}{\partial \hat{\beta_k}^s} + \left(1 - \frac{\delta^2}{V_k^{t-1} + \delta^2} \right) \frac{\partial m_k^t}{\partial \hat{\beta_k}^{(s)}}$$

with the initial condition $\partial \widetilde{m}_k^T / \partial \hat{\beta_k}^s = \partial m_k^T / \partial \hat{\beta_k}^s$. We outline the overall inference algorithm in Algorithm (2).

For prediction, since no documents are observed at test time, we initialize β values with their expected values, according to Equation 9.2 and then obtain the mean of the posterior distribution by the Kalman filter algorithm, as a standard problem. By using the learned $\boldsymbol{\pi}$ values, we could easily predict the volume of terms through Equation 9.1.

9.3 Baseline Models

Time series analysis has been long studied in many fields. Here, we discuss the possibility to employ one traditional time series tool, autoregressive model, to track the volume of terms. In univariate autoregressive model $\texttt{AR}(p)$, a response X_t can depend on its previous values, ranging from X_{t-1} to X_{t-p}:

$$X_t = w + \sum_{k=1}^{p} \pi_k X_{t-k} \tag{9.6}$$

Table 9.1: AR model on NIPS dataset

p	2007	2008	2009	Avg.
1	98.57	90.51	99.42	96.17
2	101.72	83.20	**91.06**	92.00
3	**97.66**	77.31	97.00	**90.39**
4	112.83	**75.62**	95.98	94.81
5	118.10	91.64	108.33	106.03
6	118.65	99.00	108.34	108.66
7	118.76	98.99	117.50	111.75
8	122.73	95.93	116.72	111.79
9	122.55	96.23	115.85	111.54
10	143.17	100.71	124.40	122.76

Table 9.2: AR model on ACL dataset

p	2005	2006	2007	2008	2009	Avg.
1	**131.85**	524.04	**39.57**	592.91	**126.29**	282.93
2	210.74	316.38	106.31	434.15	181.98	249.91
3	247.73	248.17	104.72	**381.84**	140.87	**224.65**
4	258.74	246.58	114.23	447.71	166.09	246.67
5	244.41	**223.99**	53.12	428.17	185.00	226.94
6	250.49	297.98	42.74	385.26	209.24	237.14
7	169.25	328.75	51.14	345.98	262.54	231.53
8	168.54	332.20	51.58	396.08	291.13	247.90
9	155.96	326.73	47.11	400.96	291.60	244.47
10	156.59	355.13	49.15	399.28	310.65	254.16

where w is a constant and $\boldsymbol{\pi}$ is a vector of coefficients. Similar to linear regression, the aim of AR(p) is to learn w and $\boldsymbol{\pi}$, as well as the optimal choice of p, sometimes. If we treat the volume of each term as X, it is obvious that the volume of terms are independent with each other. A slightly more complicated model, Multivariate AutoRegressive model MAR(p), captures the correlations between M variables and preserves the simplicity of the model:

$$\mathbf{X}_t = \mathbf{w} + \sum_{k=1}^{p} \mathbf{A}_k \mathbf{X}_{t-k} \tag{9.7}$$

where $\mathbf{X}$ and $\mathbf{w}$ are both M dimensional vectors and each $\mathbf{A}$ is a $M \times M$ matrix, encoding the correlations. Although it first seems appealing, some limitations of the model prevent it from being applied in text mining scenarios. One of the drawbacks is that the model usually requires the number of variables to be smaller than the time stamps, which is not a problem in many traditional fields (e.g., temperature and humidity over time). However, in many text corpora, we wish to track thousands, or even millions of terms (e.g., in Twitter) while the total number of time epochs to be measured is significantly smaller (e.g., in year, months, days). In that case, it is impossible to solve the Equation 9.7, according to Neumaier and Schneider [149]. Therefore, we do not use `MAR` in our experiments.

The second baseline used in experiments is Latent Dirichlet Allocation (`LDA`) [28]. We run `LDA` for the whole dataset. For each time epoch t, we obtain empirical topic distributions on t, $\boldsymbol{\beta}^t$. For each term v, we treat $\beta_{(,v)}$ as features and $Y_v^{(t)}$ as the response, building a regression model on them. Note, this model is unrealistic because in reality, we cannot obtain empirical topic distributions from the test set due to the fact that no documents should be observed from the test set. However, we include this model in the experiments for the purpose to show that topic representations can help volume prediction. A more realistic state-of-the-art model, `DTM`, is also used in the experiments. Like our model, $\boldsymbol{\beta}$ values on the test time epoch are estimated by the Kalman filter algorithm. Similar to `LDA`, the topic distributions obtained by `DTM` are treated as features and we build a regression model based upon these features. The regression model used in experiments is Support Vector Regression (`SVR`), implemented in `libSVM`[2].

[2]http://www.csie.ntu.edu.tw/~cjlin/libsvm/

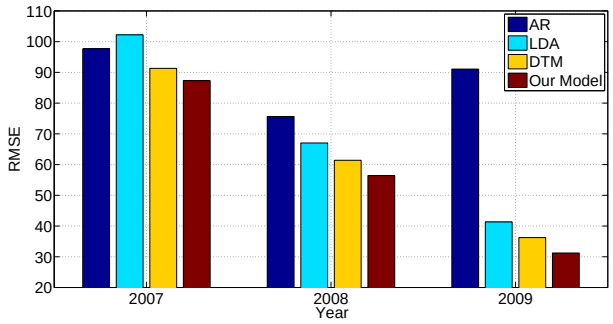

Figure 9.2: Performance comparison on the NIPS dataset.

9.4 Experiments

Two datasets of scientific papers are used in our experiments. One is from the NIPS conference series. We downloaded all electronic copies of papers from online proceedings[3] and converted into text format using pdftotext. We tokenize the converted files and keep the terms with frequency larger than 10, resulting in to 38,029 distinct terms and 4,360 papers in total, spanning 24 years. The second dataset is from the 2009 release of The ACL Anthology[4], consisting of text format of papers published in the community of computational linguistics. This dataset has 14,590 papers with 74,189 distinct terms (frequency more than 10), ranging over 37 years. Both datasets have timelines that are long enough such that some topics have changed over time.

The major evaluation measure is of course the accuracy of the predicted volume of terms. In this work, we denote the estimated volume of term v at time stamp t as $\hat{Y}_v^{(t)}$. Therefore, we measure the estimation error by calculating the Root Mean Square Error (RMSE) between estimated values and real values:

$$\text{RMSE}_t = \sqrt{\frac{1}{V}\sum_v \left(\hat{Y}_v^{(t)} - Y_v^{(t)}\right)^2}$$

[3] http://books.nips.cc/
[4] http://clair.si.umich.edu/clair/anthology/

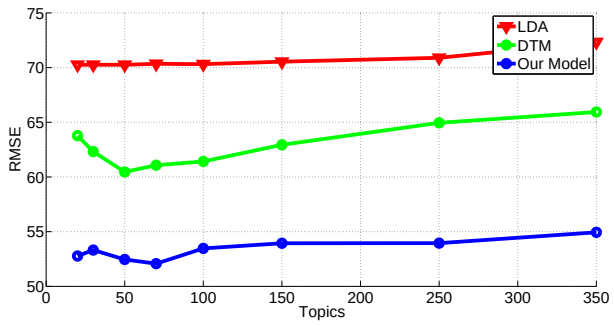

Figure 9.3: Performance comparison of different K on the NIPS dataset.

For both datasets, we adapt an "incremental" evaluation process, mimicking real application scenarios. In order to predict the volume at time t, we use the documents in all possible previous years for training. We sequentially train and test the model in multiple years and average the RMSE over these time periods. We conduct experiments on the last three years for the NIPS dataset and the last five years for the ACL dataset. For hyper-parameters, $\boldsymbol{\alpha}$ is set to $50/K$, δ^2 is set to 0.1 and $\hat{\delta}^2$ is set to 1.0, similar as [26], for all experiments.

9.4.1 Volume Prediction

As discussed in Section 9.3, the first baseline we consider is the AR model for terms. In our case, we essentially build an AR model for each term. Rather than choosing the optimal p by some criteria, such as Bayesian information criterion (BIC)[5] or Akaike information criterion (AIC)[6], we simply show the predictive performance by varying p values. Therefore, it is possible that the optimal p value is out of the ranges demonstrated here. The results for the AR model on the NIPS dataset are shown in Table 9.1 and the results on the ACL dataset are shown in Table 9.2, where the optimal performance is in bold. Several conclusions can be made regarding these results. First, for both datasets, the optimal performance is

[5]http://en.wikipedia.org/wiki/Bayesian_information_criterion
[6]http://en.wikipedia.org/wiki/Akaike_information_criterion

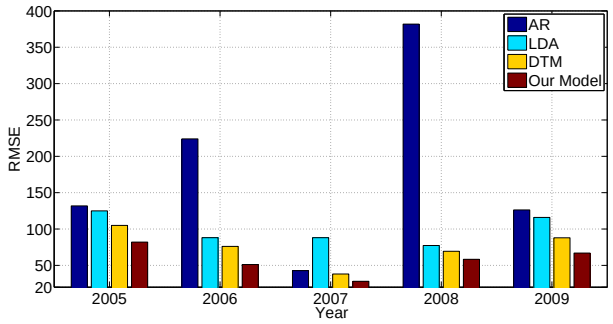

Figure 9.4: Performance comparison on the ACL dataset.

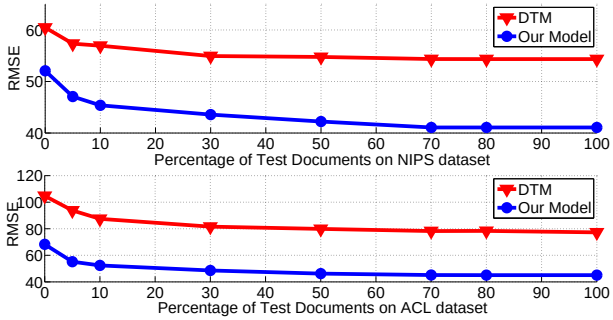

Figure 9.5: Performance when a fraction of the test documents is provided to the model.

not always obtained on $p = 1$, when the volume of terms only depends on the previous year. On average, $p = 3$ gives optimal performance on both datasets, meaning that the volume of terms in the year t depends on the previous three years. For the NIPS dataset, after the optimal point, the performance decreases as p increases, which indicates that for the AR model, no additional advantages can be obtained if we consider higher order dependencies on this particular dataset. This observation might also indicate that the latent relationships among terms, essentially topics, may change over time. Some new terms are introduced and some old concepts are outdated. For the ACL dataset, this is more complicated since the performance fluctuates significantly as p varies. Unlike the the NIPS dataset in which performance is relatively consistent over the recent three years,

predictive performance on the ACL dataset differs significantly from year to year.

We run LDA, DTM and our model on both datasets while varying the number of topics, K. The results for the NIPS and the ACL datasets are shown in Figures 9.2 where the best RMSE values achieved by each model are shown for the last three years, and 9.4 where the best RMSE values achieved by each model are shown for the last five years, respectively. For each model, we only report its best performance. In addition, for both datasets, we also compare these models to the best performance achieved by the AR model. Note, as we mentioned before, LDA is unrealistic since $\boldsymbol{\beta}$ values for the test years are from test documents while in reality these values should be estimated from the past, assuming no documents observed in these test years. However, the purpose of showing the results from plain LDA is to demonstrate that the volume predictive performance can be greatly improved by treating topic probabilities as features if we can obtain them "correctly". For DTM and our model, these $\boldsymbol{\beta}$ values are estimated by the Kalman filter algorithm, mentioned in Section 9.1, which do not depend on the test documents at all. The first observation is that the overall performance is significantly improved over the AR model, in general. LDA is usually, but not always, better than AR in terms of average performance. For DTM and our model, which both consider temporal smoothing on topics, the performance is consistently better than both LDA and AR. Our model is also better than DTM on both datasets not only in terms of average performance but also in terms of performance on individual years.

In order to better understand the performance of topic models, we plot the performance on different K values averaged over the test years for the NIPS dataset in Figure 9.3. It is clear that performance is relatively stable compared to the AR model, where it is sensitive to the p value, shown in Table 9.1. However, for all models, as K increases, the performance slightly decreases, indicating that a higher value of K may lead models to over-fit. In any case, optimal performance is obtained from 50-70 topics for DTM and our model, which seems reasonable since NIPS is a relatively small research community and

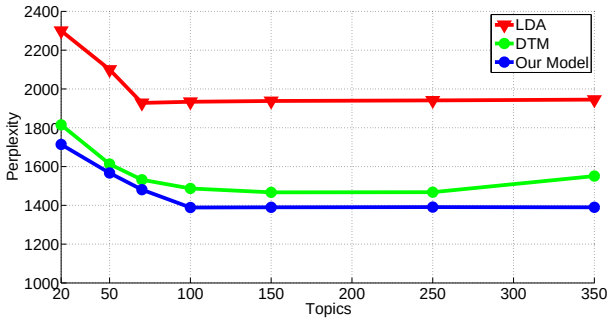

Figure 9.6: Perplexity comparison on NIPS dataset.

the topics are consistent over consecutive years. Similar conclusions can also be made for the ACL dataset.

Since DTM and our model prediction are performed on the year in which no documents are observed, it may be interesting to see whether performance would be improved if we partially observe the test documents. We pick the best K from the above experiments and feed a given fraction of test documents in a particular year to both models. The results are shown in Figure 9.5. As expected, performance improves on both datasets for both models if we observe partial data. However, when around 30% to 50% of test documents are observed, performance stabilizes.

9.4.2 Temporal Perplexity

Although we argue that perplexity may not be an appropriate evaluation method for temporal topic models, or for topic models in general, we still provide a comparison of perplexity between LDA, DTM and our model. Note, the performance on perplexity might be misleading because this measure is to evaluate how words in the documents can be assessed. Therefore, we perform the standard steps to calculate perplexity on documents in test years. As mentioned earlier, the real performance of these models should be considered when test documents are not available and how reliably the models can predict

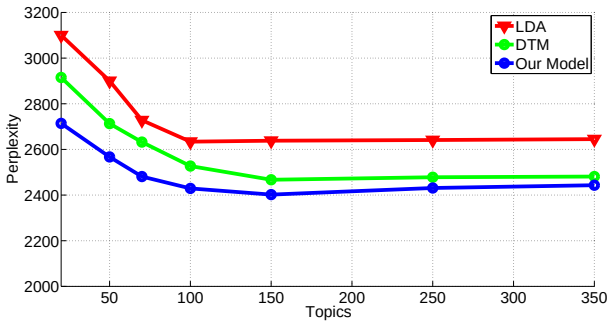

Figure 9.7: Perplexity comparison on ACL dataset.

the response variables, not words. We show perplexity on the NIPS and ACL datasets in Figures 9.6 and 9.7, respectively. Overall, the perplexity values of DTM and our model are lower than LDA, for different K values, which confirms the observations in [26, 196]. In addition, perplexity decreases as K increases in general, indicating that a larger K may explain words better. However, the difference of perplexity between DTM and our model is relatively small, compared to the volume predictive performance. This is not unexpected because our model shares the same "generative" process for words in documents as DTM. Therefore, this observation also confirms that perplexity may not be appropriate to truly reflect the performance of different models, in terms of the tasks we care about. However, we do believe that a thorough study of the relationships of perplexity and the performance of third-party tasks for topic models is needed.

9.5 Summary

In this chapter, we introduced a real-world task—tracking the volume of terms—to which temporal topic models can be applied. We proposed a new type of topic model incorporating the volumes of terms into the temporal dynamics of topics and directly optimize for

the task. We combined state-space models and the volume of terms in a supervised learning fashion which enables us to effectively predict the volume in the future. The volumes of latent topics are by-products of our model, demonstrating the superiority of utilizing temporal topic models over traditional time-series tools (e.g., autoregressive models) to tackle this kind of problem. The proposed model can be further extended with arbitrary word-level features which are evolving over time. We presented the results of applying the model to two datasets with long time periods and showed its effectiveness over non-trivial baselines. Future work might include the adoption of recently developed online variational inference algorithms [86] to our model, enabling the processing of large scale datasets.

9.6 Bibliographic Notes

In this section, we review three directions of related work. First, we summarize all up-to-date topic models which try to incorporate temporal dynamics into the model. Then, we discuss the evaluation of these models and the potential to apply them in real-world applications. In the end, we present the attempts to embed side-information, or features into topic models.

To incorporate temporal dynamics into topic models, many models have been proposed. Note, as we mentioned, these attempts are general-purpose models, meaning that no real-world tasks are explicitly addressed. In general, all these models fall into two categories. The models in the first category do not impose a global distribution assumption about how topics evolve over time. In other words, these models assume that topics change over time depending on their previous conditions, effectively making "Markovian assumptions". The examples in this category are Dynamic Topic Model (`DTM`), proposed by Blei and Lafferty [26] and Continuous Time Dynamic Topic Models (`cDTM`), proposed by Wang et al. [196], embedding state-space models into topic models. Our work is inspired by

Table 9.3: Evaluation on Temporal Topic Models

(Temporal) Perplexity	[26, 148, 196, 199, 224, 6, 103, 111]
Timestamp Prediction	[201, 196, 111]
Classification/Clustering	[224]
Ad-Hoc	[201, 199, 224]

this type of model. The second category of models usually imposes a global distribution of temporal dynamics. For instance, Wang et al. [201] introduce a beta distribution over timestamps and incorporate it into the standard topic model. Masada et al. [140] assume a Gaussian distribution over the whole time-line of topics. Although these models are proposed under different contexts, the drawback of this category is that the distributional assumption is hard to justify. Based on the two basic categories, other extensions are proposed. For example, Nallapati et al. [148] and Iwata et al. [103] focus on the problem of modeling topic spreading on timelines with multiple resolutions, namely how topics can be organized in a hierarchical way over time.

As in traditional topic models, the effectiveness of temporal topic models is difficult to evaluate in general. This is partly because these models are introduced without considering any tasks, making the process of evaluating them on third-party tasks ad-hoc. Due to a lack of evaluation tasks, comprehensive comparisons between models are seldom conducted. In order to better illustrate how temporal topic models have been evaluated, we show them in Table 9.3, according to the evaluation methods mentioned in papers. It is clear that temporal perplexity is a popular evaluation method. However, as pointed out in [31], perplexity may not have correlations with the quality (e.g., coherence) of latent topics. In addition, little is known, both theoretically and empirically, that a model achieving lower perplexity will perform better on real-world applications which we care about. Besides perplexity, several papers proposed some ad-hoc evaluation methods (named under "Ad-hoc" in the table) to demonstrate the potential capabilities of their models, such as

the coherence of topics measured by K-L divergence, where these methods are not shared by other papers and are also not really task-driven. Nearly all papers show "anecdotal examples" of what kind of topics are found over time.

Since our model can be considered as an extension to incorporate side information, or features into topic models, we also review other similar attempts. Basically, two kinds of side information might be considered: document-level features and word-level features. For document-level features, models are proposed (e.g., [27, 126, 229, 164]) to incorporate them either conditioned on latent topic assignments or conditioned on per-document hyper-parameters. Either maximum conditional learning or max-margin learning is employed for inference. For word-level features, a recently proposed model [158] introduce a method to embed arbitrary word-level features. Unlike the ones for document-level features, this model is not a fully generative model and therefore we cannot easily infer these feature values.

Chapter 10

Topic Modeling with Geographical Information

In previous two chapters, we explored how temporal information can be handled through topic modeling. These techniques are demonstrated on two specific applications on capturing temporal dynamics of topics. Here, we change our focus to utilizing geographical information. Micro-blogging services such as Twitter, Tumblr and Weibo, have become very important tools for online users to share breaking news and interesting stories. They are even used for organizing flash mobs and protest groups. For example, Twitter was used extensively in a number of events and emergencies, ranging from elections, earthquakes and tsunamis to playing an instrumental role in facilitating political upheavals in the Middle East.

Key Questions: In addition to its use as a content sharing platform, micro-blogging services like Twitter, along with other location sharing services such as Foursquare, Gowalla, and Facebook Places are nowadays supporting location services. That is, users are able to specify their location in messages, either explicitly, by letting users choose their place, or implicitly, by enabling geo-tagging functionality. This presents an exciting

opportunity to answer a range of questions:

1. How is information created and shared in different geographic locations? What is the inherent geographic variability of content?

2. What are the spatial and linguistic characteristics of people? How does this vary across regions?

3. What is a good model for human mobility? Can we discover patterns in users' usage of micro-blogging services.

There exists a considerable body of research addressing these issues [142, 198, 67, 50, 49]. However, the analysis of data still poses a considerable challenge due to its *size* and due to the *integration* of a range of different attributes. To our knowledge this is the first attempt to address both scale, location and language modeling in an integrated fashion. That is, we customize the model to be sufficiently sparse to allow for a large scale in terms of users and locations. Furthermore, we design an accurate and scalable inference algorithm.

Our algorithm allows us to discover language patterns and to extract users' interests from geo-tagged messages. We achieve this thanks to (and despite of) the sheer amount of data and the diversity of language variations used on Twitter. In addition, there are many factors to influence the language used in a tweet with a particular location. For example, words used in a tweet certainly depend on the author and the location where the tweet is written.

A user in New York City might be interested in entirely different matters compared to a user in Beijing. Moreover, the choice of words is clearly influenced by the topic of the tweet. Finally, location specific language will cause the same event to be reported quite differently in different locations (e.g. a soccer game between Brazil and Italy being reported quite differently in those two countries). Thus, different geographical regions

have different language variations and topics have different chances of being discussed in these regions.

It turns out that users tend to appear only in a handful of geographic locations [50]. This is useful in improving location accuracy in estimates. The arising challenge is how to best integrate all these strands of information into a single model.

Prior Work: Previous research effort falls into two groups: Some work only models certain aspects of the problem described above while ignoring the remainder. For instance [219] investigated how location information can be used to better understand patterns in social photo sharing services. A Gaussian mixture model and a probabilistic topic model are combined to learn clusters of locations and latent topics. However, no regional language models are learned and user preferences are also not taken into account. Thus, models developed for such data are usually limited and cannot easily be applied to content-rich social media. Similarly [50] proposed a two component Gaussian mixture model to study the mobility of users in a number of location sharing services. However, their model does not incorporate content at all. At the other end of the spectrum we find rather complex models, however, without the ability to scale to industrial size. For instance [67] propose a model to predict locations of users in Twitter. Their model has a global topic matrix and each region has different variation of this matrix. However, the inference algorithm is complex. Furthermore, the problem of over-parametrization makes it nontrivial to perform inference accurately. Furthermore, previous models ignore user preferences.

Our Contribution: We propose a model that is both flexible enough to embed all reasonable components of content and geographical locations, as well as user preference modeling. Moreover, it scales to real-world datasets to handle millions of documents and users.

In this chapter, we address the problem of modeling geographical topical patterns on Twitter by introducing a novel sparse generative model. It utilizes both statistical

topic models and sparse coding techniques to provide a principled method for uncovering different language patterns and common interests shared across the world. Our approach is vital for applications such as user profiling, content recommendation and topic tracking and the method can be easily extended in a number of ways. We show that interesting topics can be identified by the model and we demonstrate its effectiveness on the task of predicting locations of new messages and outperform non-trivial baselines. The main contributions are as follows:

- An additive generative model of content and locations that incorporates multiple facets of micro-blogging environments in an integral fashion.
- Sparse coding techniques and Bayesian treatments are smoothly embedded in our modeling, resulting in an efficient and effective implementation.
- Our model outperforms several state-of-the-art algorithms in the task of location predictions and it demonstrates interesting patterns in real-world datasets.

The chapter is organized as follows. In Section 10.9 we will briefly discuss some recent related work in terms of geographical modeling in micro-blogging environments. In Section 10.2 we proceed with detailed description of the proposed model with implementation notes. In Section 10.7 we compare our model with several state-of-the-art algorithms in a number of tasks and demonstrate its effectiveness. Finally, we conclude in Section 10.8 with discussions and future work. We now introduce our model that addresses the problems raised in the previous sections. We start with an overview of the basic components in Section 10.1 by discussing generative models without explicit switch variables. This allows us to describe the basic aspects of our model in Section 10.2. In order to learn more discriminative features, in Section 10.3, we impose L_1 penalty on certain parts of our model, resulting in a sparse modeling approach. For geographical modeling, non-informative prior

distributions are discussed in Section 10.5. More implementation details follow in Section 10.6.

10.1 Preliminaries

Our model is closely related to the Sparse Additive Generative model (`SAGE`). The basic idea of the `SAGE` model is that the outcome variable is generated by the mixture of all components without any explicit indicator variable. The key difference to traditional mixture models is that the mixture occurs not in terms of the expectation parameters (i.e. the distribution) but in terms of the natural parameters of the exponential family model. Such a model has the advantage that it can easily take a large number of aspects into account without having to infer a complex indicator variable distinguishing the set of causes.

To be more concrete, we take language modeling as an example. Suppose we have a vocabulary $\mathcal{V}$ where each term v is generated by a background language model $\boldsymbol{\phi}_0$, a per-user background language model $\boldsymbol{\phi}_u$ and a regional language model $\boldsymbol{\phi}_g$. A conventional mixture model would attempt to represent the joint influence of the three components by a linear combination of the associated densities. Denote by $p(v \mid \boldsymbol{\phi})$ an exponential family model of the form

$$p(v \mid \boldsymbol{\phi}) = \exp\Big(\boldsymbol{\phi}_v - g(\boldsymbol{\phi})\Big) \text{ where } g(\boldsymbol{\phi}) = \log \sum_v \exp\Big(\boldsymbol{\phi}_v\Big)$$

Here $g(\boldsymbol{\theta})$ is often referred to as the log-partition function as it ensures that the distribution is properly normalized. In particular for the discrete distribution $\phi(v|\boldsymbol{\phi})$ is well-defined for all choices of $\boldsymbol{\phi}$. We now combine the factors via

$$P(v \mid \boldsymbol{\phi}_0, \boldsymbol{\phi}_u, \boldsymbol{\phi}_g) := p(v \mid \boldsymbol{\phi}_0 + \boldsymbol{\phi}_u + \boldsymbol{\phi}_g) \tag{10.1}$$

Table 10.1: Notation

Symbol	Size	Usage
$\boldsymbol{\eta}^0$	$1 \times \mathbb{R}$	global region distribution
$\boldsymbol{\eta}^{\text{user}}$	$\mathbb{U} \times \mathbb{R}$	user-dependent region distribution
$\boldsymbol{\theta}^0$	$1 \times \mathbb{K}$	global topic distribution
$\boldsymbol{\theta}^{\text{geo}}$	$\mathbb{R} \times \mathbb{K}$	region-dependent topic distribution
$\boldsymbol{\theta}^{\text{user}}$	$\mathbb{U} \times \mathbb{K}$	user-dependent topic distribution
$\boldsymbol{\phi}^0$	$1 \times \mathbb{V}$	global term distribution
$\boldsymbol{\phi}^{\text{geo}}$	$\mathbb{R} \times \mathbb{V}$	region-dependent term distribution
$\boldsymbol{\Pi}$	$\mathbb{K} \times \mathbb{V}$	a global topic matrix
$\boldsymbol{\mu}$	$\mathbb{R}^2$	mean location of a latent region
$\boldsymbol{\Sigma}$	$\mathbb{R}^{2\times 2}$	covariance matrix of a latent region

Unlike in traditional topic models, the formalism above does not require an indicator variable to specify which component to use in generating v. In addition to additive modeling, different language models can be constructed in such a way as to incorporate more discriminative terms. More specifically, in our model we choose $\boldsymbol{\phi}_0$ to denote the (baseline) log frequency of v in the dataset while other components are used to model the differences between the baseline and the background model. This idea is explored in [231, 65] to model topics. Here, we extend it to model regions and topics jointly and to propose an efficient inference procedure.

10.2 Model Description

We start the discussion with some notations in our model. Each tweet $d = \{\mathbf{w}_d, \mathbf{l}_d, u_d\}$ consists of three parts: Here $\mathbf{w}_d$ is the word vector for the tweet, following a simple bag of word assumption, $\mathbf{l}_d$ is a real-valued pair $\mathbf{l}_d = \{l_0, l_1\}$, representing the latitude and longitude where this tweet is written and u_d is the user id for the author of the tweet. For simplicity, we assume that all the tweets in our dataset are generated by a fixed vocabulary $\mathcal{V}$ and a fixed user base $\mathcal{U}$. Moreover, we assume that the geographical locations have been

clustered into R latent regions. Each region $r \in R$ is characterized by a mean location $\boldsymbol{\mu}_r$ and a covariance matrix $\boldsymbol{\Sigma}_r$. We assume that there are three types of language models: a) a background language model $\boldsymbol{\phi}^0$, b) a per-region language model $\boldsymbol{\phi}^{\text{geo}}$ and c) a topical language model $\boldsymbol{\Pi}$. All these language models are over the vocabulary $\mathcal{V}$. Each tweet is influenced by these three factors simultaneously. Before describing the generative process of our model, on a high level, our model encodes the following intuitions:

- Words used in a tweet depend on both the location and topic of the tweet.
- Different geographical regions have different language variations. Topics have different chances to be discussed in different regions (e.g. bullfights in India are unlikely to occur; likewise Spaniards are unlikely to discuss Divali).
- Users tend to appear in a handful geographical locations.

For each tweet, the model generates the location, the topic and terms in the tweet consecutively. In our model, all locations are categorized into R latent regions. For each tweet, we first choose from which latent region this tweet is written. To generate the region index r, we utilize a multinomial model as follows:

$$P\Big(r \,|\, \boldsymbol{\eta}^0, \boldsymbol{\eta}_u^{\text{user}}\Big) = p\Big(r \,|\, \boldsymbol{\eta}^0 + \boldsymbol{\eta}_u^{\text{user}}\Big) \tag{10.2}$$

Here $\boldsymbol{\eta}_0$ is a global distribution over latent regions and $\boldsymbol{\eta}_u$ is a user dependent distribution over latent regions for user u. Each location $\mathbf{l}_d$ is drawn from a latent region r by a region-dependent multivariate normal distribution

$$\mathbf{l}_d \sim \mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r). \tag{10.3}$$

Once the region and the location is generated, a topic z is selected dependent on both the latent region and the author of tweet:

$$P\Big(z \,|\, \boldsymbol{\theta}^0, \boldsymbol{\theta}_u^{\text{user}}, \boldsymbol{\theta}_r^{\text{geo}}\Big) = p\Big(z \,|\, \boldsymbol{\theta}_j^0 + \boldsymbol{\theta}_{u,j}^{\text{user}} + \boldsymbol{\theta}_{r,j}^{\text{geo}}\Big) \tag{10.4}$$

Here $\boldsymbol{\theta}^0$ is a global distribution over topics, $\boldsymbol{\theta}_u^{\text{user}}$ is a user-dependent distribution over topics and $\boldsymbol{\theta}_r^{\text{geo}}$ is a regional distribution over topics. The intuition is that the topic is heavily influenced where this tweet is written and user preferences. After generating the topic index z each word w in the tweet is generated by drawing from the aggregate distribution:

$$P\Big(w \,|\, z, \boldsymbol{\phi}^0, \boldsymbol{\phi}_r^{\text{geo}}, \boldsymbol{\Pi}_z\Big) = p\Big(w \,|\, \boldsymbol{\phi}^0 + \boldsymbol{\phi}_r^{\text{geo}} + \boldsymbol{\Pi}_{z_d}\Big). \tag{10.5}$$

In this case $\boldsymbol{\phi}^0$ parametrizes a global distribution over terms, $\boldsymbol{\phi}^{\text{geo}}$ describes the a region-dependence and $\boldsymbol{\Pi} \in \mathbb{R}^{\mathbb{K}\times\mathbb{V}}$ is a topic matrix where each row is a distribution over terms.

With the above specification the generative story for a single tweet d can be expressed as follows:

- Draw a latent region index

$$r_d \sim p(r_d \,|\, \boldsymbol{\eta}^0 + \boldsymbol{\eta}_u^{\text{user}})$$

- Draw a topic index

$$z_d \sim p(z_d \,|\, \boldsymbol{\theta}^0 + \boldsymbol{\theta}_u^{\text{user}} + \boldsymbol{\theta}_r^{\text{geo}})$$

- Draw a location

$$\mathbf{l}_d = \{l_0, l_1\} \sim \mathcal{N}(\boldsymbol{\mu}_r, \boldsymbol{\Sigma}_r)$$

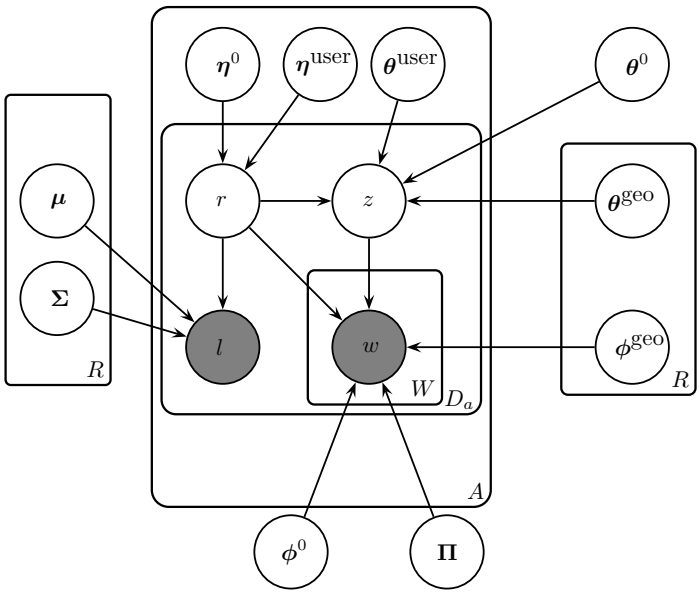

Figure 10.1: A graphical representation of our model

- For each token w in $\mathbf{w}_d$ draw

$$w \sim p(w \mid \boldsymbol{\phi}^0 + \boldsymbol{\phi}_r^{\text{geo}}, \boldsymbol{\Pi}_{z_d})$$

This generative process applies to all tweets in the corpus. The graphical representation of the generation process is shown in Figure 10.1.

10.3 Sparse Modeling

As discussed in Section 10.1, the benefit of our approach is to learn discriminative features from data, rather than obtaining redundant ones in different components of the model. In order to achieve this goal, we also impose prior distributions over certain parts of our model. More specifically, for the following components in the model, we impose zero-mean Laplace distributions.

The rationale is that users in certain regions are likely to draw their words either from a location independent distribution or from a *small*, i.e. sparse corpus of additional terms which are more prevalent in a given location rather than globally. Likewise, we assume that topics consist of a background distribution of generic words plus a sparse set of additional words which are characteristic for the particular topic. Note that we do *not* require these words to be unique. That is, the word "jaguar" might for instance be more prevalent in the "animals" and in the "cars" topic. However, we do not expect it to be prevalent in a large number of topics beyond what a background language model would indicate. We have

$$
\begin{aligned}
\boldsymbol{\eta}_r^0 &\sim \mathcal{L}(0, \omega^0) & \boldsymbol{\eta}_{u,r}^{\text{user}} &\sim \mathcal{L}(0, \omega_u) & & \\
\boldsymbol{\theta}_z^{\text{geo}} &\sim \mathcal{L}(0, \lambda_l) & \boldsymbol{\theta}_{u,z}^{\text{user}} &\sim \mathcal{L}(0, \lambda_u) & \boldsymbol{\theta}_{r,z}^{\text{geo}} &\sim \mathcal{L}(0, \lambda_r) \\
\boldsymbol{\phi}_v^0 &\sim \mathcal{L}(0, \psi^0) & \boldsymbol{\phi}_{r,v}^{\text{geo}} &\sim \mathcal{L}(0, \psi_l) & & \\
\boldsymbol{\Pi}_{z,v} &\sim \mathcal{L}(0, \psi_t) & & & &
\end{aligned}
$$

where $\mathcal{L}(\mu, b)$ is a Laplace distribution with mean μ and scale parameter b. A zero-mean Laplace prior has the same effect as placing an L_1 regularizer on these components, resulting in a sparse solution to the model. Here, a sparse modeling approach does not only encourage more discriminative features to be learned, but also leads to a more efficient learning algorithm, which will be introduced below. We use `ISTA` [20] algorithm to do sparse optimization in our work. Note that besides Laplace distributions used in this chapter, other distributions could be employed, too. For instance using a normal distribution as prior on all elements amounts to a latent Gaussian process induced by the parameters.

10.4 Inference Algorithm

Before we proceed with the inference algorithm, we introduce the following shorthands to simplify our notation:

$$P(z_d = k \,|\, \boldsymbol{\theta}^0, \boldsymbol{\theta}_u^{\text{user}}, \boldsymbol{\theta}_r^{\text{geo}}) = \boldsymbol{\alpha}_{u,r,k}$$
$$P(w = v \,|\, z_d, \boldsymbol{\phi}^0, \boldsymbol{\phi}_r^{\text{geo}}, \boldsymbol{\Pi}) = \boldsymbol{\beta}_{r,z,v}$$
$$P(r = t \,|\, \boldsymbol{\eta}^0, \boldsymbol{\eta}_u^{\text{user}}) = \boldsymbol{\rho}_{u,t}$$

We treat topic assignments z and latent region assignments r as latent variables and all other variables as model parameters. A mixture between EM and a Monte Carlo sampler is utilized to effectively learn all parameters for the model along the lines of [193]. In the E-step, we sample latent region assignments and topic assignments by fixing all other parameters by Gibbs sampling. In the M-step, we optimize model parameters by fixing all latent region assignments and topic assignments. We iterate this until convergence.

More specifically, in the E-step, we iteratively draw latent region assignments and topic assignments for all tweets. For each tweet, a latent region r is firstly drawn from the following distribution, conditioned on the old topic assignments:

$$r \sim P(\mathbf{l}_d \,|\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \times \boldsymbol{\rho}_{u,j} \times \boldsymbol{\alpha}_{u,j,k} \times \prod_{i=1}^{N_d} \boldsymbol{\beta}_{j,k,v} \tag{10.6}$$

where $P(\mathbf{l}_d \,|\, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ is the pdf function for a multivariate normal distribution and k is the old topic assignment. After r is sampled, we sample the topic assignment z for the same tweet, conditioned on the newly sampled r:

$$z \sim \boldsymbol{\alpha}_{u,r,k} \times \prod_{i=1}^{N_d} \boldsymbol{\beta}_{r,z,v} \tag{10.7}$$

where r is the new region index. In the M-step, we maximize the log likelihood of the model with respect to model parameters by fixing all region and topic assignments obtained in the E-step. For geographical modeling, the maximum likelihood estimation (MLE) of parameters can be obtained in closed form:

$$\boldsymbol{\mu}_j = \bar{N}_j = \frac{1}{\#(d,j)} \sum_{d=1}^{D} \mathbb{I}(r_d = j)\mathbf{l}_d \tag{10.8}$$

$$\boldsymbol{\Sigma}_j = S_j = \frac{1}{\#(d,j)-1} \sum_{d=1}^{D} (\mathbf{l}_d - \boldsymbol{\mu}_j)^T (\mathbf{l}_d - \boldsymbol{\mu}_j) \tag{10.9}$$

where $\#(d,j)$ is the number of tweets assigned to region j. Indeed, $\boldsymbol{\mu}_j$ is set to the sample mean and $\boldsymbol{\Sigma}_j$ is set to the sample variance. For other parameters, unfortunately, no closed-form solutions exist. Therefore, we adopt gradient-based optimization methods to maximize the likelihood. Let L be the likelihood of the model. The gradients of model parameters can be obtained as follows. For $\boldsymbol{\eta}^0$ and $\boldsymbol{\eta}^{\text{user}}$, we have:

$$\begin{aligned} \partial \boldsymbol{\eta}_t^0(L) &= \sum_{u=1}^{U} d(u,t) - \sum_{u=1}^{U} d(u)\boldsymbol{\rho}_{u,t} \\ \partial \boldsymbol{\eta}_{u,t}^{\text{user}}(L) &= d(u,t) - d(u)\boldsymbol{\rho}_{u,t} \end{aligned} \tag{10.10}$$

where $d(u,t)$ is the number of tweets produced by user u are assigned to the region t and $d(u)$ is the total number of tweets generated by user u. For the global topic distribution

$\boldsymbol{\theta}^0$, user topic distributions $\boldsymbol{\theta}^{\text{user}}$ and regional topic distributions $\boldsymbol{\theta}^{\text{geo}}$, we have:

$$\partial \boldsymbol{\theta}_k^0(L) = \sum_{u=1}^{U} d(u,k) - \sum_{u=1}^{U}\sum_{t=1}^{R} d(u,t)\boldsymbol{\alpha}_{u,t,k} \tag{10.11}$$

$$\partial \boldsymbol{\theta}_{u,k}^{\text{user}}(L) = d(u,k) - \sum_{t=1}^{R} d(u,t)\boldsymbol{\alpha}_{u,t,k} \tag{10.12}$$

$$\partial \boldsymbol{\theta}_{t,k}^{\text{geo}}(L) = \sum_{u=1}^{U} d(u,t,k) - \sum_{u=1}^{U} d(u,t)\boldsymbol{\alpha}_{u,t,k} \tag{10.13}$$

where $d(u,k)$ is the number of tweets produced by user u assigned to the topic k and $d(u,t,k)$ is the number of tweets written by the user u in the region t assigned to the topic k. For the global language model $\boldsymbol{\phi}^0$, regional language models $\boldsymbol{\phi}^{\text{geo}}$ and topical language models $\boldsymbol{\Pi}$, we have:

$$\partial \boldsymbol{\phi}_v^0(L) = \sum_{t=1}^{R} n(t,v) - \sum_{t=1}^{R}\sum_{k=1}^{K} n(t,k)\boldsymbol{\beta}_{t,k,v} \tag{10.14}$$

$$\partial \boldsymbol{\phi}_{t,v}^{\text{geo}}(L) = n(t,v) - \sum_{k=1}^{K} n(t,k)\boldsymbol{\beta}_{t,k,v} \tag{10.15}$$

$$\partial \boldsymbol{\Pi}_{k,v}(L) = \sum_{t=1}^{R} n(t,k,v) - \sum_{t=1}^{R} n(t,k)\boldsymbol{\beta}_{t,k,v} \tag{10.16}$$

where $n(d,v)$ is the number of times term v appearing in tweet d, $n(t,k)$ is the number of terms associated to the topic k in region t, $n(t,v)$ is the number of times term v appearing region t, $n(t,k,v)$ is the number of terms v assigned to the topic k appearing in the region t. These gradients have an intuitive interpretation as the difference of the true counts and their expected counts.

10.5 Geograpical Location Modeling

In the previous section, we use a point estimate of regional means and covariance matrices in each M-step based on samples obtained in the E-step. However, this process is not very stable since only one sample of regional assignments for each tweet is taken into account. One way to reduce this instability would be to draw multiple samples per tweet and to use a set of samples for estimation purposes. However, this would introduce an inner loop in the E-step for each tweet, thus significantly increasing sampling time.

Instead, we apply a Bayesian treatment to mean vectors and covariance matrices and do not estimate them explicitly in M-step. The standard practice in multivariate normal distribution is to endow them with a set of conjugate parameters, that is, with a Gauss-Wishart prior. This is computationally expensive.

A cheaper (and equally reliable) approach is to place a non-informative Jeffrey's prior over the values of the mean parameters, that is

$$\boldsymbol{\mu} \sim \text{Unif.}$$

and a Jeffrey's distribution over the values of the covariance matrices to penalize large covariance matrices:

$$P(\boldsymbol{\Sigma}) \propto |\boldsymbol{\Sigma}|^{-(3/2)}.$$

The same treatment is also used in [7, 76].

By imposing these prior distributions, we can effectively integrate out $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, resulting in a collapsed Gibbs sampler for locations, similar to [79]. More specifically, we sample r from the following distribution:

$$r \sim T\Big(\bar{N}_r, S_r \frac{(n+1)}{n(n-2)}, n-2\Big)\boldsymbol{\rho}_{u,j}\boldsymbol{\alpha}_{u,j,k}\prod_{i=1}^{N_d}\boldsymbol{\beta}_{j,k,v} \tag{10.17}$$

Here $T(a, b, n)$ is a multivariate Student-T distribution with the location as a, the scale matrix as b and n degree of freedom. Here, $\bar{N}_r$ and S_r are sample mean and sample respectively, as defined in 10.8. Sampling r does not require us to re-estimate the values of mean and covariance matrix in the M-step and hence reduce the computation cost of the inference algorithm.

10.6 Implementation Notes

Several implementation notes warrant a detailed discussion here. Firstly, the bottleneck of sampling z is to evaluate many exponential functions as we expand Equation (10.7):

$$\frac{\exp\Big(\boldsymbol{\theta}_k^0 + \boldsymbol{\theta}_{u,k}^{\text{user}} + \boldsymbol{\theta}_{r,k}^{\text{geo}}\Big)}{\sum_{i=1}^{K} \exp\Big(\boldsymbol{\theta}_i^0 + \boldsymbol{\theta}_{u,i}^{\text{user}} + \boldsymbol{\theta}_{r,i}^{\text{geo}}\Big)} \prod_{i=1}^{N_d} \frac{\exp\Big(\boldsymbol{\phi}_{w_i}^0 + \boldsymbol{\phi}_{r,w_i}^{\text{geo}} + \boldsymbol{\Pi}_{k,w_i}\Big)}{\sum_{j=1}^{V} \exp\Big(\boldsymbol{\phi}_j^0 + \boldsymbol{\phi}_{r,j}^{\text{geo}} + \boldsymbol{\Pi}_{k,j}\Big)}$$

The key to speed up the sampling procedure here is to reduce the number of exponential functions to be evaluated. We re-write the above equation as:

$$\begin{aligned}\exp\Big[\boldsymbol{\theta}_k^0 + \boldsymbol{\theta}_{u,k}^{\text{user}} + \boldsymbol{\theta}_{r,k}^{\text{geo}} + \sum_{i=1}^{N_d}\Big(\boldsymbol{\phi}_{w_i}^0 + \boldsymbol{\phi}_{r,w_i}^{\text{geo}} + \boldsymbol{\Pi}_{k,w_i}\Big) \\ - \log \sum_{i=1}^{K} \exp\Big(\boldsymbol{\theta}_i^0 + \boldsymbol{\theta}_{u,i}^{\text{user}} + \boldsymbol{\theta}_{r,i}^{\text{geo}}\Big) \\ - N_d \log \sum_{j=1}^{V} \exp\Big(\boldsymbol{\phi}_j^0 + \boldsymbol{\phi}_{r,j}^{\text{geo}} + \boldsymbol{\Pi}_{k,j}\Big)\Big]\end{aligned}$$

The logarithm of a sum of components can be efficiently computed as $\log \sum_i \exp(x_i) = m + \log[\sum_i \exp(x_i - m)]$ where m is the maximum element in x_i and can be cached since they are constant in the E-step. Therefore, we only need to calculate one exponential function for sampling z per tweet, which significantly reduces the computational cost.

The second technique to speed up the inference algorithm is to efficiently calculate

gradients 10.14, 10.11, and 10.10. A naïve calculation would lead to a very inefficient implementation. Taking the gradients of $\mathbf{\Pi}$ as an example, the expanded form of gradients is as follows:

$$\sum_{t=1}^{R} n(t,k,v) - \sum_{t=1}^{R} n(t,k) \frac{\exp(\boldsymbol{\phi}_v^0 + \phi_{t,v}^{\text{geo}} + \mathbf{\Pi}_{k,v})}{\sum_{i=1}^{V} \exp(\boldsymbol{\phi}_i^0 + \boldsymbol{\phi}_{t,i}^{\text{geo}} + \mathbf{\Pi}_{k,i})}$$

where the second part of the gradients, which is the expected counts, requires the calculation for all the possible combinations of topics and latent regions. However, because of sparse modeling in Section (10.3), we can effectively calculate the second parts by utilizing the sparsity of the model as follows:

$$\begin{aligned}
& n(k,v) - \exp(\boldsymbol{\phi}_v^0) \sum_{t=1}^{R} n(t,k) \frac{1}{C_{t,k}} \\
& - \sum_{t=1}^{R} n(t,k) \frac{1}{C_{t,k}} \exp(\boldsymbol{\phi}_v^0) \Big[\exp(\boldsymbol{\phi}_{t,v}^{\text{geo}}) - 1) \Big] \\
& - \sum_{t=1}^{R} n(t,k) \frac{1}{C_{t,k}} \exp(\boldsymbol{\phi}_v^0) \exp(\boldsymbol{\phi}_{t,v}^{\text{geo}}) \Big[\exp(\mathbf{\Pi}_{k,v}) - 1 \Big]
\end{aligned}$$

where $C_{t,k} = \sum_{i=1}^{V} \exp(\boldsymbol{\phi}_i^0 + \boldsymbol{\phi}_{t,i}^{\text{geo}} + \mathbf{\Pi}_{k,i})$. The gradients are decomposed into three parts. The first part is a global term for all terms and therefore can be calculated once and cached. The second part only exists for those $\boldsymbol{\phi}_{t,v}^{\text{geo}}$ are not zero. Similarly, the third part is non-zero only when both $\boldsymbol{\phi}_{t,v}^{\text{geo}}$ and $\mathbf{\Pi}_{k,v}$ are not zero. Thus, if we employ a reasonable L_1 regularizer on both regional and topical language models, most of those elements would be driven to zero and therefore the second and third parts can be very efficiently calculated. Similar decomposition also works for other gradients.

The last but not the least important technique is how to initialize the model. Different initialization values of parameters can lead to significantly different results. Here, we use the following initialization steps. Again, taking language models as an example, we firstly initialize $\boldsymbol{\phi}^0$ as log frequencies of terms in the whole corpus and $\boldsymbol{\phi}_r^{\text{geo}}$ as log frequencies

of terms in region r minus the same term in ϕ^0. Then, we initialize $\mathbf{\Pi}$ as all zero and optimize over $\mathbf{\Pi}$ by fixing ϕ^0 and ϕ^{geo}. Similar strategy can be also applied to $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ values. For latent regions, we initialize them by a K-Means algorithm.

10.7 Experiments

In this section, we demonstrate the effectiveness of our model on real-world datasets. We compare our model with several state-of-the-art models. Our dataset is a sample of the Twitter Firehose stream[1], issued to Yahoo!. In Twitter, two types of location information are associated to tweets: 1) geographical locations and 2) Twitter Places[2]. For geographical locations, each tweet is associated to a real-valued latitude and longitude vector. For Twitter Places, we convert them into real-valued latitudes and longitudes. After doing this, we remove all tweets without locations. We also preprocess all the remaining tweets by detecting whether a tweet is in English. This step is done by a dictionary based method. We randomly sample 10,000 users from the dataset, with their full set of tweets between January 2011 and May 2011, resulting 573,203 distinct tweets. The size of the dataset is significantly larger than the ones used in some similar studies (e.g, [67, 219]).

10.7.1 Location Prediction

In addition to demonstrating that our model can discover interesting topics and users' geographical patterns, we also wish to show that our model can be used in a quantitative fashion. Here, we focus on the task of location prediction for tweets. Differing from the work done by Eisenstein et al. [67] where their aim is to predict the location for a user and the way they defined the location of a user may not be very appropriate (the first location

[1]https://dev.twitter.com/docs/streaming-api/methods
[2]http://blog.twitter.com/2010/06/twitter-places-more-context-for-your.html

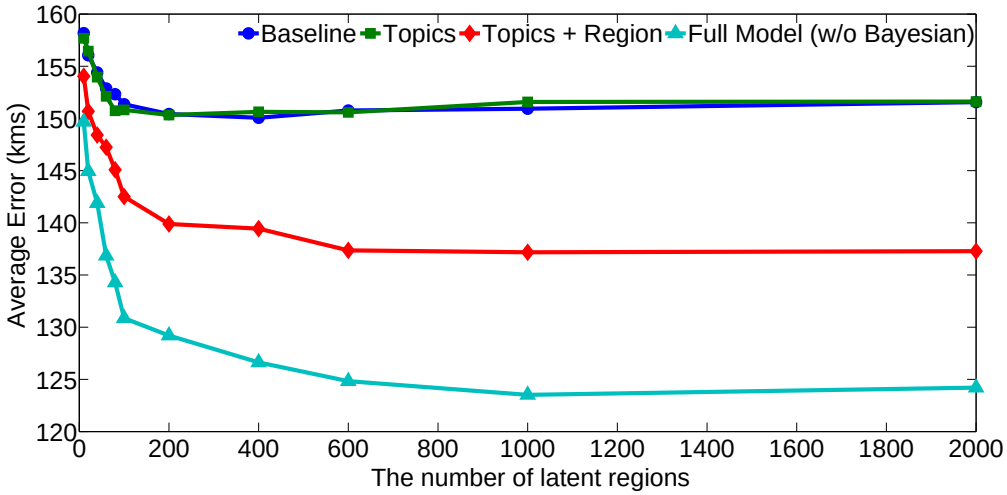

Figure 10.2: The comparison of location prediction on `Yahoo!` dataset.

shown in their dataset), our goal is to predict the location for each new tweet, based on the words used in the tweet and its authors' information. Based on our statistics, only 1% $\sim$ 2% of tweets have either geographical locations (including Twitter Places) explicitly attached, meaning that we cannot easily locate a majority of tweets. However, it has been shown (e.g., [50, 49]) that geographical locations can be used to predict users' behaviors and uncover users' interests and therefore it is potentially invaluable for many perspectives, such as behavior targeting and online advertisements. In addition to our dataset, we also apply our model to an open source datasest[3], denoted as `CMU` dataset, and compare the best reported results.

Evaluation Metric: For each new tweet, we predict its location as $\hat{\mathbf{l}}_d$. We calculate the Euclidean distance between predicted value and the true location and average them over the whole test set $\frac{1}{N}\sum \text{Dis}(\hat{\mathbf{l}}_d, \mathbf{l}_d)$ where $\text{Dis}(a, b)$ is the Euclidean distance function and N is the total number of tweets in the test set.

[3]http://www.ark.cs.cmu.edu/GeoText/

Baselines: The following methods are used as baselines in our dataset to compare with the full model proposed in Section (10.2).

- Yin et al. [219]: Their method is essentially to have a global set of topics shared across all latent regions. There is no regional language models in the model. Besides, no user level preferences are learned in the model. The prediction is done by two steps: 1) choosing the region index that can maximize the test tweet likelihood, and 2) use the mean location of the region as the predicted location. We re-implemented their method in our work. This method is denoted as `Baseline`.

- Our model without $\boldsymbol{\phi}^{\text{geo}}$, $\boldsymbol{\eta}^{\text{user}}$ and $\boldsymbol{\theta}^{\text{user}}$: This is essentially very similar to `Baseline`. The only difference is that `Baseline` is under PLSA formalism and our model is in `SAGE` formalism. We denote this method as `Topics`.

- Our model without $\boldsymbol{\eta}^{\text{user}}$ and $\boldsymbol{\theta}^{\text{user}}$: This variation of our model can learn regional language models while user preferences are still missing here. We denote this method as `Topics + Region`.

For the comparison on the `CMU` dataset, we compare with:

- Eisenstein et al. [67]: The model is to learn a base topic matrix that can be shared across all latent regions and a different topic matrix as the regional variation for each latent region. No user level preferences are learned in the model. The best reported results are used in the experiments.

- Eisenstein et al. [65]: The original `SAGE` paper. The best reported results are used in the experiments.

- Wing and Baldridge [210]: Their method is essentially to learn regional language models per explicit regions. The best reported results are used in the experiments.

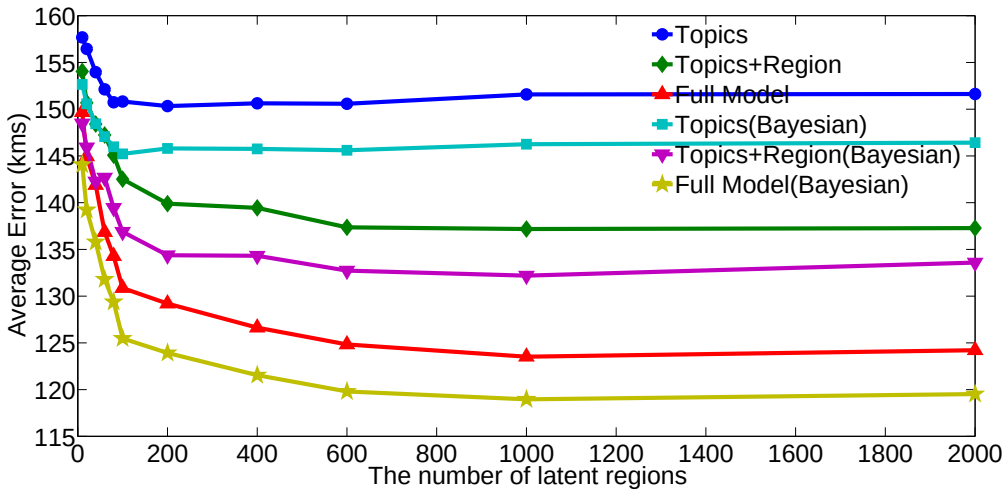

Figure 10.3: The comparison of non-Bayesian models and Bayesian models.

For our model, the prediction is conducted in two steps. Firstly, a region index that can maximize the likelihood of test tweet is chosen. Next, the mean location of the corresponding region is used as the predicted location. For Bayesian treatment of geographical modeling discussed in Section (10.5), the mean vectors are estimated after the whole inference algorithm finishes.

Experimental Results: Firstly, we show the basic comparison between our model and other baselines discussed above on the `Yahoo!` dataset. The results are shown in Figure 10.2 where the X-axis is the number of latent regions and Y-axis is the average Euclidean distance in kilometers (kms) between predicted locations and true locations. In this experiment, we fix the number of topics to 50 for all models. For all models, we adopt a five-fold cross validation setting. The numbers reported here are averaged across different folds. One major impression is that the average error decreases as the number of latent regions increases, although it becomes flat after 500 latent regions. This makes sense because we predict the locations based on the mean locations of latent regions.

Therefore, the more regions the model has, the more flexible the prediction would be. As we discussed above, `Topics` method is very similar to `Baseline` method and therefore, not very surprisingly, the performance of these two models is approximately the same. For `Topics + Region` model, the performance is significantly better over `Baseline` model and `Topics` model. The main reason might be that regional language models learn special terms for different regions and therefore these terms become discriminative when we perform location predictions. Moreover, our sparse modeling approach also contributes to learned discriminative terms in regional language models. By incorporating user regional preferences ($\boldsymbol{\eta}^{\text{user}}$), our full model performs the best on the `Yahoo!` dataset. This partially validates that users might have stable mobility patterns in their usage of micro-blogging environments and therefore we can learn this pattern through their historical content. Indeed, Cho et al. [50] found that users who frequently use location sharing services demonstrate surprisingly stable patterns and they successfully used a two-component Gaussian mixture model to predict users' locations in the future. Note that the full model used in this experiment is the one **without** Bayesian geographical modeling that is discussed in Section 10.5.

The next set of experiments is to show whether the Bayesian treatment of geographical modeling can lead to additional improvements of predication performance. As we previously discussed, non-Bayesian modeling in locations may lead to unstable results. The experimental setting follows the one used above and results on the task of location prediction on `Yahoo!` dataset are shown in Figure (10.3) where the X-axis is the number of latent regions and Y-axis is the average Euclidean distance in kilometers (kms) between predicted locations and true locations. Two observations can be made from the figure. Firstly, all models with Bayesian modeling lead to significantly improvements over their non-Bayesian counterparts. The second observation is that, although Bayesian modeling can improve the performance, major improvements still comes from whether certain

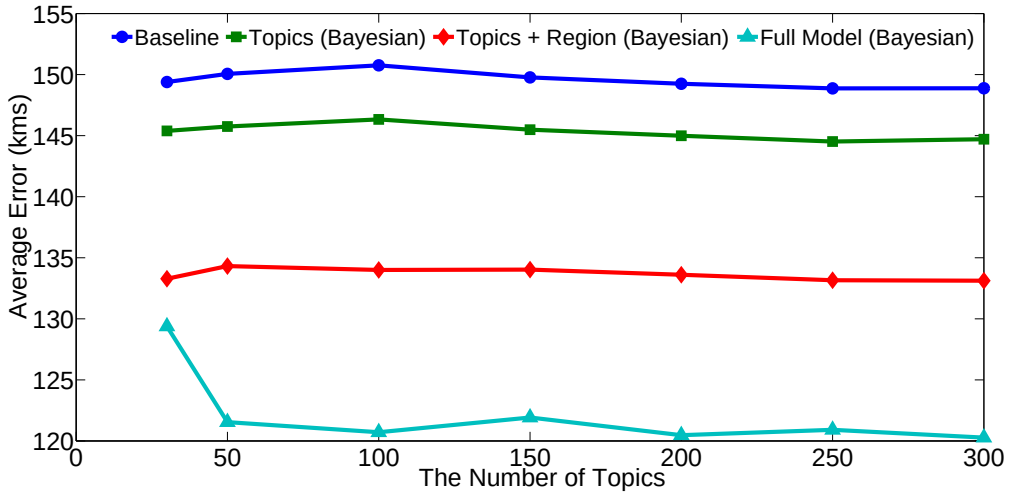

Figure 10.4: The comparison of models with different number of topics.

components are "on" or "off". In short, Bayesian modeling in locations enjoys better predictive performance and a more efficient inference algorithm, as discussed in previous sections.

All previous experiments are the ones with fixed topics and different latent regions. Here we show how the predictive performance varies for different number of topics. The basic setting remains the same as the previous two sets of experiments and the results are shown in Figure 10.4 by fixing the number of latent regions (as 400) on `Yahoo!` dataset. The X-axis is the number of topics and Y-axis is the average Euclidean distance in kilometers (kms) between predicted locations and true locations. The main observation is that the performance does not change too much as the number of topics varies. As we mentioned before, all these models make predictions based on the mean vectors of latent regions. Therefore, a fixed number of regions will limit the predictive power of these models and hence the performance is sort of bounded in a range. In other words, enlarging the number of topics does not give models the flexibility to learn regions well.

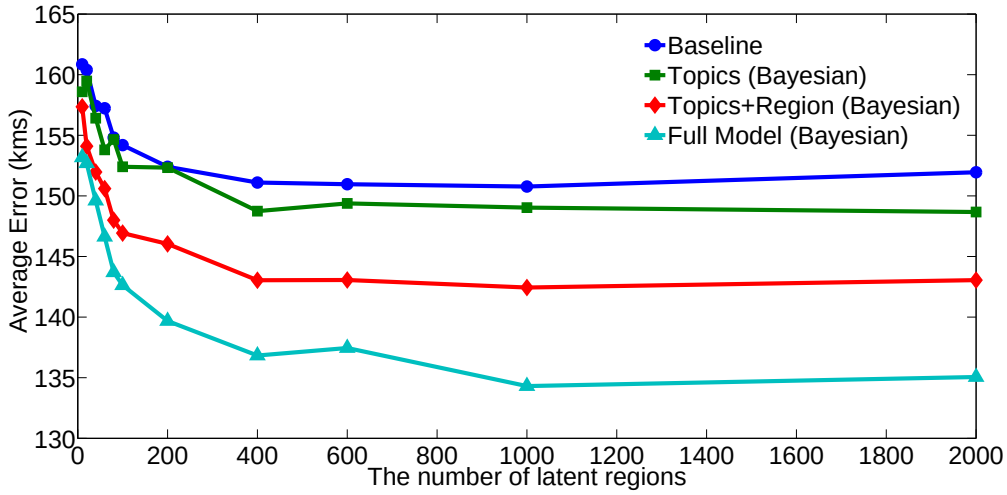

Figure 10.5: The comparison of models with randomly selected users on `Yahoo!` dataset.

Another interesting experiment is not to randomly sample tweets but randomly sample users. In this setting, all users in the test set are never shown in the training set and therefore we do not have sufficient user preference data. This setting might be more realistic in Twitter because the majority of users never use geo-related features and hence it is highly likely that some users will adopt this feature in the future. In order to effectively predict locations, we use the following strategy to learn a "prior" distribution for users. Taking $\boldsymbol{\eta}^{\text{user}}$ as an example, since the test user is not in our training set, we optimize over $\boldsymbol{\eta}^{\text{user}}$ by fixing all other parameters on the fly. Therefore, the obtained values for this user is essentially the prior regional distribution for this user, without any tweets observed. After having this prior distribution, we can effectively predict locations as usual. We do this optimization for users on the fly for all other user-related parameters. The results are shown in Figure 10.5 where the X-axis is the number of latent regions and Y-axis is the average Euclidean distance in kilometers (kms) between predicted locations and true locations.. The main observation from the figure is that the performance from all models is significantly worse than the experiments with randomly selected tweets. This partially

# of latent regions	[[67]]	[[210]]	[[65]]	Topics	Topics + Region	Full Model
10	494	479	501	540.60	481.58	449.45
20	494	479	501	522.18	446.03	420.83
40	494	479	501	513.06	414.95	395.13
60	494	479	501	507.37	410.09	380.04
80	494	479	501	499.42	408.38	374.01
100	494	479	501	498.94	407.78	372.99

Table 10.2: Comparison of models on CMU dataset. All numbers are kilometers.

validates that all these models suffer from certain difficulties for "new" users and "new" content. However, the relative improvement of performance remains the same as previous experiments, suggesting that our model can learn reasonable prior distributions for users, in order to achieve better predictive performance.

For the CMU dataset, we download their dataset and run our model on it. Note that previous models (e.g., [67, 210]) are designed to predict the locations for users. In our case, we can do finer grained predictions on tweet level. To make fair comparisons, two strategies can be applied here: 1) obtain the predicted location for each tweet and take the mean locations over them and 2) obtain the dominant region index for tweets by the same user and use the mean value for it as the prediction. In our experiments, we have tried both strategies and found no significant difference between them. Therefore, we only report the results from the first strategy. The results are shown in Table (10.2). For [67, 210, 65], the median number reported in the paper is used. We do not re-run their models and only report numbers from corresponding papers. Firstly, we see that our full model outperforms all previous models significantly. In addition, as the number of latent regions increases, the predictive performance increases, which also validates the results in our Yahoo! dataset. Here is some analysis why our model outperforms others. For [67] and [65], they used a topic-variation matrix per region, which might be too expensive to be applied over a large number of regions while the authors in those papers found that

Location with Top Ranked Terms
United States – > New York – > Brooklyn
brooklyn ave flatbush avenue mta prospect 5th #brooklyn spotlight carroll bushwick museum broadway madison vanderbilt coney slope eastern subway new york pkwy #viernesnayobon #mets otsego greenwich starbucks
United States – > California – >San Francisco
sfo francisco san airport international millbrae terminal flight burlingame bart mateo boarding bayshore telecommute landed heading bay airlines united bound flying #sfo camino groupon caltrain moon tsa baggage california engineer valley
United States – > Pennsylvania – > Philadelphia
philadelphia #philadelphia phl #jobs market others #job street philly walnut septa chestnut the cherry sansom arch spruce citizens locust btw temple pennsylvania rittenhouse passyunk bitlyetq7a6 bookrenters pike international
United Kingdom – > England – >London
winds lhr hounslow terminal the cloudy mph ickenham bath heathrow temperature airport car only airways uxbridge sun splendid fair london british lounge tothers harmondsworth speedbird whens for stars day flight dominos navigation brunel
Australia – > New South Wales – > Sydney
sydney #sydney bondi george street mascot domestic syd surry station cnr platforms harbour darlinghurst qantas hoteloxford eddy haymarket terminal wales australia chalmers uts pitt #marketing junction darling centre #citijobs citigroup druitt

Table 10.3: Examples of ϕ^{geo}.

their model peaks at around 50 regions and 10 topics and the predictive performance deteriorates otherwise for excessive number of parameters, resulting in over-fitting. In our case, we use global topics and background topics to factor out common words. In addition, we use two signals: regional topic distribution and regional word unigrams. For [67, 65], their model has a single location for all tweets per user. On the contrary, our model assumes that each user has a distribution over regions and each tweet is associated with a region, thus we can accommodate user movements. Also, their models used a two-stage training which does not enable the language model to influence how many regions are needed. However, we use a joint training procedure for both regions and topics and we re-sample the user regions in our training phase where their models assume that regions assignments are given at the first place.

10.7.2 Qualitative Study

In this section, we take one run of our full model on `Yahoo!` dataset as an example to demonstrate what kinds of topics can be obtained. Firstly, we show some samples of regional language models. As we see in the previous section, these language models play a vital role in location predictions. Since in our model, regions are latent variables and do not correspond to cities or regions in the real-world. It might be difficult to demonstrate

Entertainments
lady bieber album music beats artist video listen itunes apple produced movies #bieber lol new songs
Sports
yankees match nba football giants wow win winner game weekend horse #nba
Politics
obama election middle east china uprising egypt russian tunisia #egypt afghanistan people eu

Table 10.4: Examples of $\mathbf{\Pi}$, global topic matrix.

topics. Here, we assign the mean vectors of latent regions to nearest existing cities and manually pick 5 cities as an example, shown in Table 10.3. The terms are top ranked terms in each language model.. Terms are the ones with largest magnitudes in ϕ^{geo}. It is very interesting to see that most top ranked terms are actually the name of these locations. Remember that our method is fully unsupervised. In addition, we can see that top ranked terms in different regions vary significantly. Another interesting observation is that users tend to tweet with their locations when they are in airports. This can be seen in region "United States $->$ California $->$ San Francisco" and "United Kingdom $->$ England $->$ London". In addition to geographical language models, we also show some examples from the global topic matrix $\mathbf{\Pi}$. These language models are designed so that broader topics will be captured here. The examples are shown in Table 10.4. The terms are top ranked terms in each language model. Again, these topics are manually picked and the "title" of these topics is assigned by the authors of the paper since these topics are learned without any explicit labels. We can see that these topics are relatively broad, compared to regional language models and widely discussed across regions. Some topics might have captured recent unrest in the Middle East.

10.8 Summary

In this chapter, we address the problem of modeling geographical topical patterns on Twitter by introducing a novel sparse generative model, which utilizes both statistical topic models and sparse coding techniques to provide a principled method for uncovering different language patterns and common interests shared across the world. Our approach is vital for applications such as behavior targeting, user profiling, content recommendation and topic tracking and the method can be easily extended in a number of ways. We show that interesting topics can be identified by the model and we demonstrate its effectiveness on the task of predicting locations of new messages and outperform non-trivial baselines. Main contributions of this work include a) a sparse additive model of content and locations that incorporate multiple facets of micro-blogging environments without switch variables, b) sparse coding techniques and Bayesian treatments are smoothly embedded in our modeling, resulting in an efficient and effective implementation and c) outperforms several state-of-the-art algorithms in the task of location predictions and demonstrate interesting patterns from real-world datasets. For future work, we wish to model human mobility explicitly by introducing user level regional components. In addition, temporal factors should also be considered for the task of location prediction.

10.9 Bibliographic Notes

We briefly review two lines of related research. The first is a range of papers which use geographical language modeling in general while the second is a set of works which are specifically tuned for Twitter data. We are particularly interested in models and approaches that combine geographical modeling and language modeling to discover topics from geographical regions. We summarize some of representative work here:

- Mei et al. [142] propose a model based on Probabilistic Latent Semantic Indexing

(PLSA) [88]. It assumes that each word is either drawn from a universal background topic or from a location and time dependent language model. Inference is performed via EM. However, the mixture coefficients between the background topic and other spatio-temporal topics ones is tuned manually. Since the model uses PLSA, no prior distribution is (or could be) assumed. Evaluation is carried out by showing anecdotal results.

- Later, Wang et al. [198] introduce a fully Bayesian generative model to incorporate locations. Rather than working with real latitudes and longitudes, they have a fixed number of region labels and they assume that each term is associated with a location label. For each word in a document, a topic assignment is first generated according to a multinomial distribution. Then the term and the location are generated dependent on this topic assignment, according to two different multinomial distributions. The inference is performed by Variational EM. Again the evaluation is limited to anecdotal results.

- Sizov [182] propose a similar model to [198]. Rather than using a multinomial distribution to generate locations they replace it with two Gaussian distributions for generating latitude and longitude respectively. For inference, this work uses Gibbs Sampling and the evaluation is done by showing anecdotal results, by measuring Deviation Information Criteria (a model complexity criterion similar to BIC), as well as classification accuracy using manually labeled data. One of the drawbacks of the work is that they only use data from Flickr restricted to the greater London area.

- Hao et al. [82] propose a model built upon Wang et al. [198]. However, they introduce the notion of global topics and local topics where more general terms are grouped into global topics and terms related to local events going to local topics. The inference

is performed by Gibbs Sampling. Hao et al. [82] evaluate their model based on anecdotal results and some heuristic measurements.

- Yin et al. [219] propose a model is similar in spirit to Eisenstein et al. [67]. The terms and the location of a particular document are generated by a latent region. The location is generated from a region by a normal distribution and the region is sampled from a multinomial distribution. The prior is also placed into the model, however the inference is done by MAP-style EM rather than a fully Bayesian fashion. The model is evaluated using perplexity and by showing anecdotal results.

- Wing and Baldridge [210] use an even simpler approach where documents are assigned to geodesic grids and thus a supervised learning method is utilized, essentially yielding to build naïve Bayes classifiers on geodesic grids.

Although there exists such attempts of modeling language patterns and geographical locations, *most prior work does not consider users at all.*

A second line of work covers models directly designed to work on Twitter data. For instance, Eisenstein et al. [67] propose a model utilizing the correlations between global and local topics. In their model, each author is assigned a latent region variable and an observed GPS location. Terms and the actual GPS location are both conditioned on the latent region variable. The topics to generate terms are local topics, which are derived from global topics. The inference is done by Variational EM and the evaluation is done by measuring the accuracy of predicted location and showing anecdotal results. Finally, Cho et al. [50] studied the problem of human mobility in location sharing services. Their findings include that users tend to appear in a very limited number of places (e.g., office and home). They demonstrated that it might be effective enough to use a two component Gaussian mixture model to estimate users' locations.

It has been an active research area to incorporate different information sources into

topic modeling. For example, Chemudugunta et al. [43] propose a method to combine corpus-wide topics and document-specific language patterns together by using a "switch" variable for each term in the document, becoming a popular scheme in topic modeling literature. We use a "switch-free" approach in this work and therefore reduce the number of variables used in the model. Last, for general patterns and analysis of social location sharing services, please refer to Cheng et al. [49].

Chapter 11

Conclusion and Future Work

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

11.1 Summary

Online conversational media has gained popularity in recent years due to the emergence and advances of social media services, bringing users across the globe to have conversations and share information more effectively and efficiently. It attracts a great amount of research on mining and understanding useful patterns from online conversational media. In this thesis, we mainly focus on two fundamental problems, information filtering and users' interests modeling – trying to provide a guideline for social media researchers and practitioners to build information systems that can serve the increasingly dynamic user needs. On high level, in terms of methodologies, the contributions of this thesis to the aspect of information filtering include:

- a comprehensive study on how users' authority impact the performance of retrieval tasks in online conversational media, showing 10% improvement over state-of-the-art counterparts in some tasks.

- a generic method to predict the popularity of a message in an online conversational media, the first study in this line of research with a thorough discussion of effectiveness of a wide range of features.

- a probabilistic framework to predict personal user decisions of messages in an online conversational media, including application scenarios in an online professional social network and a micro-blogging service. The framework utilizes collaborative filtering, meta-information and content information to estimate the likelihood of an action, resulting in an 15% to 20% improvement over other state-of-the-art algorithms.

- a comprehensive empirical study of how topic modeling results can be used in information filtering, a first study in this line of research, which sheds the lights on the importance of topic modeling to the research of onilne conversational media in general.

Using these contributions, one can easily build information filtering systems for similar environments. The last contribution leads to the investigation of embedding different meta-information to topic modeling, in the hope of developing more effective models to reveal hidden patterns in online conversational media. In this thesis, the contributions of topic modeling include:

- a topic model that can directly handle the term volumes and predict them in a time series manner, a first work to address the evaluation problem of temporal topic models, achieving significant improvement over non-trivial standard time-series models.

- a topic model that utilizes multiple information sources to discover topics with temporal dynamics, a first work to tackle this practical work and outline a framework to deal with similar tasks.

- a general topic model to incorporate geographical information associated with conversations to discover hidden language patterns in online conversational media.

All these models can be easily used for building information systems that need to utilize topics.

In addition to new methodologies and frameworks, some of the methods and the techniques developed in this dissertation improve state-of-the-art algorithms in building personalization and recommendation systems in online conversational media. In particular, in this thesis, we have improved the state-of-the-art in the following scenarios:

- For **filtering question answer pairs from CQA portals**, we explore the problem of filtering question answering content from discussion boards and divide it into two subtasks: identifying question–related first posts and finding potential answers in subsequent responses within the corresponding threads. We address both subtasks as classification problems and choose several content–based and non-content based features and carefully compare them individually and also in combinations. We do not use any service or dataset–specific heuristics or features (like the rank of users) in our classification model; therefore our approach should be usable in any discussion board. We compare our approach with previous methods and show 10% improvements in experimental results.

- For **filtering popular messages in micro-blogging services**, we cast the problem into a classification framework and build classifiers with positive and negative examples of messages which will be retweeted in the future and which contain URLs which are shared in the future. To build such classifiers, we investigate a wide spectrum of features to determine which ones can be successfully used as predictors of popularity, including the content and topical information of messages, graph

structural properties of users, temporal dynamics of popular messages and meta-information of users and messages as well. Our experiments are conducted on two massive real-world datasets and the results suggest that we can successfully predict whether a message will be popular or not and its volume with 20% more F_1 compared to non-trivial baselines.

- For **filtering relevant messages for each individual in micro-blogging services**, we study the problem of predicting whether a user will take actions towards a message in micro-blogging environment. Our method can be easily extended to model multiple types of users' decisions as well. We propose Co-Factorization Machines (`CoFM`), which deal with two (multiple) aspects of the dataset where each aspect is a separate `FM`. This type of model can easily predict user decisions while modeling user interests through content at the same time. With this tool, we apply Factorization Machines to text data with constraints. Thus, the resulting method can mimic state-of-the-art topic models and yet benefit from the efficiency of a simpler form of modeling. We apply our proposed methods to the problem of modeling personal decision making in Twitter and explore a wide range of features, revealing which types of features contribute to the predictive modeling and how content information can help with the prediction, resulting 10% improvement over existing state-of-the-art models.

- For **tracking temporal trends**, we propose a new type of topic model incorporating the volume of terms into the temporal dynamics of topics and directly optimize for the task. Unlike existing models in which trends are either latent variables or not considered at all and thus are difficult to apply in practice, we combine state-space models with term volumes in a supervised learning fashion which enables us to effectively predict volumes in the future, even without new documents. In addition, it

is straightforward to obtain the volumes of latent topics as a by-product of our model, demonstrating the superiority of utilizing temporal topic models over traditional time-series tools (e.g., autoregressive models) to tackle this kind of problem. The proposed model can be further extended with arbitrary word-level features which are evolving over time. We present the results of applying the model to two datasets with long time periods and show its effectiveness by improving non-trivial baselines over 15% to 20% in terms of prediction accuracy.

- For **modeling temporal dyanmics on multiple data sources**, we extend topic models by allowing each text stream to have both local and shared topics. Also, we associate each topic with a time-dependent function that characterizes its popularity over time. By combining the two models, we effectively model temporal dynamics of multiple correlated text streams in a unified framework. The new model can easily discover common and uncommon topics from multiple text collections with their temporal dynamics. The proposed method is a simple and potentially scalable algorithm for mining temporal topics. We mined interesting results from Yahoo! News and Twitter obtained by applying our model.

- For **geographical information**, we propose a model that is both flexible enough to embed all reasonable components of content and geographical locations, as well as user preference modeling. Moreover, it scales to real-world datasets to handle millions of documents and users. We address the problem of modeling geographical topical patterns on Twitter by introducing a novel sparse generative model. It utilizes both statistical topic models and sparse coding techniques to provide a principled method for uncovering different language patterns and common interests shared across the world. Our approach is vital for applications such as user profiling, content recommendation and topic tracking and the method can be easily

extended in a number of ways. We show that interesting topics can be identified by the model and we demonstrate its effectiveness on the task of predicting locations of new messages and outperforms existing generative models by 20% in terms of prediction accuracy.

11.2 Future Work

There are multiple potential directions for future work, based on this thesis. Here, we present three directions:

- **Allowing in-depth understanding of conversations**: Although online conversations play a central role in this dissertation and different aspects of conversations are discussed, it is lacking an in-depth understanding of them. It is believed that traditional language processing and understanding techniques cannot be easily applied to online conversational media for the reasons discussed in the introduction of this thesis. For instance, some researchers [77, 131] have tried to conduct part-of-speech recognition tasks on Twitter, a typical online conversational media, and faced a significant challenge, compared to traditional data such as news articles. Similarly, Ritter et al. [170] proposed a generative model, trying to uncover conversational structure among users from Twitter. The model makes strong assumption on how conversations are generated and the performance is not comparable to traditional domains. In all, it is an attractive challenge to develop models for deep understanding of content in online conversational media.

- **Integrating graph-based models**: Currently, all models utilize network or graph information as features of more high-level models, such as classifiers or regression models. More complicated models have been developed for graphs or networks in a

newly emerged subfield, network science. It would be interesting to explore the possibility of bridging topic modeling, recommendation and network science, by utilizing the tools of probabilistic modeling. Currently, they are developed in separate communities and each direction may not be fully aware of the other ones. For instance, network models [151] are rarely explored in topic modeling and recommendation communities and shallow features about networks are usually calculated and used. The challenges of modeling social network enabled data have the potential to be enhanced by network models.

- **Scalable learning algorithms**: The sheer amount of data from online conversational media places a tremendous challenge for learning a meaningful model in an effective and efficient way. Many models presented in this thesis are or can be cast into graphical models. It has been an active research area to develop large-scale learning algorithms for graphical models. For instance, one direction of research is to exploit large computing clusters of commodity machines, adapting existing algorithms in such contexts. In this approach, a common case is to modify existing single machine algorithms or impose complex scheduling techniques to ensure that these algorithms can achieve reasonable results on large clusters (e.g., [150, 183, 5]). One drawback of this direction is that, some speedup techniques are model specific and the proposed architectures usually do not fit into the current standard MapReduce paradigm. An alternative direction is to develop intrinsically fast algorithms, which can scale to large datasets even on a single machine. For example, recently developed stochastic variational inference techniques for graphical models (e.g., [86, 197]) might be a promising direction to analyze large datasets with millions of data instances on a single machine. However, the drawbacks of this direction are that 1) these techniques require a non-trivial number of hyper-parameters to tune and 2) no clear vision exists for parallelizing these algorithms. In general, scalable learning

algorithms still remain a tremendous challenge for probabilistic modeling.

11.3 Conclusion

In this dissertation, we provide a comprehensive study of mining and understanding online conversational media, focusing on problems of information filtering and topic modeling. For information filetering, we develop a series of predictive models with a wide range of features to tackle tasks such as detecting question-answer pairs from community-based answering portals, identifying global popular messages from and designing personal information filtering algorithms for online professional services and micro-blogging services. For topic modeling, we develop probablistic models to incorporate temporal and geographical information into existing graphical topic models to discover interesting patterns from online social data. In both directions, we advance the state-of-the-art and demonstrate that the online conversational media can be tackled in a large scale. The future study stands on allowing deeper understanding of conversational structures and integrating network models. In addition, scalable learning algorithms will become the backbone technique to support those areas.

Bibliography

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 19–28, New York, NY, USA, 2009. ACM.

[2] D. Agarwal and B.-C. Chen. fLDA: matrix factorization through latent Dirichlet allocation. In *Proceedings of the third ACM International Conference on Web Search and Data Mining (WSDM)*, pages 91–100, New York, NY, USA, 2010. ACM.

[3] D. Agarwal, B.-C. Chen, and B. Long. Localized factor models for multi-context recommendation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 609–617, New York, NY, USA, 2011. ACM.

[4] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *Proceedings of the 1st International ACM Conference on Web Search and Web Data Mining (WSDM)*, pages 183–194, New York, NY, 2008. ACM.

[5] A. Ahmed, M. Aly, J. Gonzalez, S. Narayanamurthy, and A. J. Smola. Scalable inference in latent variable models. In *Proceedings of the fifth ACM International*

conference on Web Search and Data Mining, WSDM '12, pages 123–132, New York, NY, USA, 2012. ACM.

[6] A. Ahmed and E. P. Xing. Timeline: A dynamic hierarchical Dirichlet process model for recovering birth/death and evolution of topics in text stream. In *Proceedings of the 26th International Conference on Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 20–29, 2010.

[7] A. Ahmed, E. P. Xing, W. W. Cohen, and R. F. Murphy. Structured correspondence topic models for mining captioned figures in biological literature. In *Proceedings of KDD 2009*, pages 39–48, New York, NY, USA. ACM.

[8] A. Aji and E. Agichtein. Deconstructing interaction dynamics in knowledge sharing communities. In *International Conference on Social Computing, Behavioral Modeling, and Prediction*, pages 273–281, 2010.

[9] D. J. Aldous. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581 – 598, 1981.

[10] D. J. Aldous. Exchangeability and related topics. volume 1117 of *Lecture Notes in Mathematics*, chapter 1, pages 1–198. Springer Berlin Heidelberg, 1985.

[11] L. Alsumait, D. Barbará, J. Gentle, and C. Domeniconi. Topic significance ranking of LDA generative models. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pages 67–82, 2009.

[12] F. Antonelli and M. Sapino. A rule based approach to message board topics classification. In *Advances in Multimedia Information Systems*, pages 33–48, 2005.

[13] J. Arguello, J. L. Elsas, J. Callan, and J. G. Carbonell. Document representation and query expansion models for blog recommendation. In *Proceedings of the Second International Conference on Weblogs and Social Media (ICWSM)*, 2008.

[14] J. M. Atkinson and J. Heritage. *Structures of social action: studies in conversation analysis*. Cambridge University Press, 1984.

[15] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. Four degrees of separation. *CoRR*, abs/1111.4570, 2011.

[16] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, C. Cortes, and M. Mohri. Polynomial semantic indexing. In *NIPS*, pages 64–72, 2009.

[17] S. Balakrishnan and S. Chopra. Collaborative ranking. In *WSDM*, pages 143–152, 2012.

[18] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, Dec. 2005.

[19] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *KDD 2008*, pages 16–24, 2008.

[20] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, March 2009.

[21] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing user behavior in online social networks. In *SIGCOMM 2009*, pages 49–62.

[22] A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal. Bridging the lexical chasm: Statistical approaches to answer-finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–199, New York, NY, 2000. ACM.

[23] J. Bian, Y. Liu, E. Agichtein, and H. Zha. Finding the right facts in the crowd: factoid question answering over social media. In *WWW '08: Proc. of the 17th International Conference on World Wide Web*, pages 467–476, New York, NY, USA, 2008. ACM.

[24] J. Bian, Y. Liu, D. Zhou, E. Agichtein, and H. Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *Proc. of the 18th International Conference on World Wide Web (WWW)*, pages 51–60, 2009.

[25] D. Blei and J. Lafferty. Topic models. *Text Mining: Theory and Applications*, 2009.

[26] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 113–120, 2006.

[27] D. M. Blei and J. D. Mcauliffe. Supervised topic models. In *Advances in Neural Information Processing Systems 21*, 2007.

[28] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[29] C. Borgs, J. T. Chayes, B. Karrer, B. Meeder, R. Ravi, R. Reagans, and A. Sayedi. Game-theoretic models of information overload in social networks. In *Workshop on Algorithms and Models for the WebGraph 2010*, pages 146–161.

[30] M. Bouguessa, B. Dumoulin, and S. Wang. Identifying authoritative actors in question-answering forums: The case of Yahoo! answers. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 866–874, New York, NY, 2008. ACM.

[31] J. Boyd-Graber, J. Chang, S. Gerrish, C. Wang, and D. Blei. Reading tea leaves: How humans interpret topic models. In *Neural Information Processing Systems (NIPS)*, 2009.

[32] G. Brown and G. Yule. *Discourse analysis*. Cambridge University Press, 1983.

[33] S. Budalakoti and R. Bekkerman. Bimodal invitation-navigation fair bets model for authority identification in a social network. In *WWW 2012*, pages 709–718.

[34] B. Cao, D. Shen, K. Wang, and Q. Yang. Clickthrough log analysis by collaborative ranking. In *AAAI 2010*.

[35] Y. Cao, H. Duan, C.-Y. Lin, Y. Yu, and H.-W. Hon. Recommending questions using the MDL-based tree cut model. In *Proceeding of the 17th international conference on World Wide Web (WWW)*, pages 81–90, New York, NY, USA, 2008. ACM.

[36] V. R. Carvalho and W. W. Cohen. On the collective classification of email "speech acts". In *Proceedings of the 28th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 345–352, New York, NY, USA, 2005.

[37] V. R. Carvalho and W. W. Cohen. Improving email speech acts analysis via n-gram selection. In *Proceedings of the HLT/NAACL 2006 Analyzing Conversations in Text and Speech Workshop (ACTS)*, pages 35–41, New York City, NY, June 2006. Association for Computational Linguistics.

[38] C. Castillo, D. Donato, and A. Gionis. Estimating number of citations using author reputation. In *Proceedings of the 14th International Conference on String Processing and Information Retrieval*, pages 107–117, 2007.

[39] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available from `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[40] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27:1–27:27, May 2011.

[41] J. Chang, J. Boyd-Graber, and D. M. Blei. Connections between the lines: augmenting social networks with text. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178, New York, NY, USA, 2009. ACM.

[42] O. Chapelle and M. Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval*, 13:216–235, 2010.

[43] C. Chemudugunta, P. Smyth, and M. Steyvers. Modeling general and specific aspects of documents with a probabilistic topic model. In *NIPS*, pages 241–248, 2006.

[44] H. Chen, S. R. K. Branavan, R. Barzilay, and D. R. Karger. Global models of document structure using latent permutations. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 371–379, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[45] J. Chen, R. Nairn, and E. Chi. Speak little and well: recommending conversations in online social streams. In *CHI 2011*, pages 217–226.

[46] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and tweet: experiments on recommending content from information streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 1185–1194, New York, NY, USA, 2010.

[47] K. Chen, T. Chen, G. Zheng, O. Jin, E. Yao, and Y. Yu. Collaborative persoanlized tweet recommendation. In *SIGIR*, 2012.

[48] T. Chen, Z. Zheng, Q. Lu, W. Zhang, and Y. Yu. Feature-based matrix factorization. Technical report, Apex Data & Knowledge Management Lab, Shanghai Jiao Tong University, 2011.

[49] Z. Cheng, J. Caverlee, K. Lee, and D. Sui. Exploring millions of footprints in location sharing services. In *ICWSM 2011*.

[50] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of KDD 2011*, pages 1082–1090, New York, NY, USA. ACM.

[51] M. D. Choudhury, H. Sundaram, A. John, and D. D. Seligmann. What makes conversations interesting? Themes, participants and consequences of conversations in online social media. In *Proc. of the 17th Int'l Conf. on World Wide Web (WWW)*, pages 331–340, Apr. 2009.

[52] W. W. Cohen, V. R. Carvalho, and T. M. Mitchell. Learning to classify email into "speech acts". In D. Lin and D. Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 309–316, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[53] M. Collins, S. Dasgupta, and R. E. Schapire. A generalization of principal components analysis to the exponential family. In *Advances in Neural Information Processing Systems 14 (NIPS)*, pages 617–624, Vancouver, British Columbia, Canada, 2001.

[54] G. Cong, L. Wang, C.-Y. Lin, Y.-I. Song, and Y. Sun. Finding question-answer pairs from online forums. In *Proceedings of the 31st Annual International ACM SIGIR*

Conference on Research and Development in Information Retrieval, pages 467–474, New York, NY, 2008. ACM.

[55] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM 2008*, pages 87–94.

[56] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[57] M. De Choudhury, S. Counts, and M. Czerwinski. Identifying relevant social media content: leveraging information diversity and user cognition. In *Hypertext and Hypermedia 2011*, pages 161–170.

[58] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, Jan. 2008.

[59] N. A. Diakopoulos and D. A. Shamma. Characterizing debate performance via aggregated twitter sentiment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 1195–1198, New York, NY, USA, 2010.

[60] C. Ding, T. Li, and W. Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Comput. Stat. Data Anal.*, 52:3913–3927, April 2008.

[61] G. Doyle and C. Elkan. Accounting for burstiness in topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 281–288, 2009.

[62] H. Duan, Y. Cao, C.-Y. Lin, and Y. Yu. Searching questions by identifying question topic and question focus. In *ACL-08: HLT*, 2008.

[63] Y. Duan, L. Jiang, T. Qin, M. Zhou, and H.-Y. Shum. An empirical study on learning to rank of tweets. In *COLING 2010*, pages 295–303.

[64] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 272–279, New York, NY, USA, 2008. ACM.

[65] J. Eisenstein, A. Ahmed, and E. Xing. Sparse additive generative models of text. In *Proceedings of ICML 2011*, pages 1041–1048, New York, NY, USA, June. ACM.

[66] J. Eisenstein and R. Barzilay. Bayesian unsupervised topic segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 334–343, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

[67] J. Eisenstein, B. O'Connor, N. A. Smith, and E. P. Xing. A latent variable model for geographic lexical variation. In *Proceedings of EMNLP 2010*, pages 1277–1287, Stroudsburg, PA, USA. Association for Computational Linguistics.

[68] J. L. Elsas, J. Arguello, J. Callan, and J. G. Carbonell. Retrieval and feedback models for blog feed search. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 347–354, New York, NY, USA, 2008.

[69] J. L. Elsas and J. G. Carbonell. It pays to be picky: an evaluation of thread retrieval in online forums. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 714–715, New York, NY, USA, 2009.

[70] M. Elsner and E. Charniak. You talking to me? a corpus and algorithm for conversation disentanglement. In *Proceedings of the 2008 Meeting of the Association for Computational Linguistics (ACL)*, pages 834–842, 2008.

[71] D. Feng, E. Shaw, J. Kim, and E. Hovy. An intelligent discussion-bot for answering student queries in threaded discussions. In *Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI)*, pages 171–177, New York, NY, 2006. ACM.

[72] D. Feng, E. Shaw, J. Kim, and E. Hovy. Learning to detect conversation focus of threaded discussions. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 208–215, Morristown, NJ, 2006. Association for Computational Linguistics.

[73] M. Galassi, J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi. *GNU Scientific Library Reference Manual - Third Edition (v1.12)*. Network Theory Ltd., 2009. `http://www.gnu.org/software/gsl/`.

[74] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*, pages 176–185, 2010.

[75] J. P. Gee. *An Introduction to Discourse Analysis: Theory and Method*. Routledge, 2005.

[76] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman-Hall, 2nd edition, 2003.

[77] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter:

annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 42–47, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[78] M. Goetz, J. Leskovec, M. McGlohon, and C. Faloutsos. Modeling blog dynamics. In *International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2009.

[79] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, April 2004.

[80] Z. Gyöngyi, G. Koutrika, J. Pedersen, and H. Garcia-Molina. Questioning Yahoo! Answers. In *Proceedings of the First Workshop on Question Answering on the Web*, 2008.

[81] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, pages 10–18, 2009.

[82] Q. Hao, R. Cai, C. Wang, R. Xiao, J.-M. Yang, Y. Pang, and L. Zhang. Equip tourists with knowledge mined from travelogues. In *Proceedings of WWW 2010*, pages 401–410, New York, NY, USA. ACM.

[83] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW '02: Proceedings of the 11th International Conference on World Wide Web*, pages 517–526, New York, NY, USA, 2002. ACM.

[84] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

[85] P. D. Hoff. Multiplicative latent factor models for description and prediction of social networks. *Computational & Mathematical Organization Theory*, 15:261–272, December 2009.

[86] M. Hoffman, D. Blei, and F. Bach. Online learning for latent dirichlet allocation. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 856–864, 2010.

[87] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. of the 22nd Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 50–57, 1999.

[88] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42:177–196, 2001.

[89] L. Hong. A tutorial on probabilistic latent semantic analysis. *CoRR*, abs/1212.3900, 2012.

[90] L. Hong, A. Ahmed, S. Gurumurthy, A. J. Smola, and K. Tsioutsiouliklis. Discovering geographical topics in the Twitter stream. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pages 769–778, New York, NY, USA, 2012.

[91] L. Hong, R. Bekkerman, J. Adler, and B. D. Davison. Learning to rank social update streams. In *Proceedings of the 35th international ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 651–660, New York, NY, USA, 2012. ACM.

[92] L. Hong, O. Dan, and B. D. Davison. Predicting popular messages in Twitter. In *Proceedings of the 20th International Conference Companion on World Wide Web (WWW)*, pages 57–58, New York, NY, USA, 2011.

[93] L. Hong and B. D. Davison. A classification-based approach to question answering in discussion boards. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 171–178, New York, NY, USA, 2009.

[94] L. Hong and B. D. Davison. Empirical study of topic modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics (SOMA)*, pages 80–88, New York, NY, USA, 2010.

[95] L. Hong, B. Dom, S. Gurumurthy, and K. Tsioutsiouliklis. A time-dependent topic model for multiple text streams. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 832–840, New York, NY, USA, 2011.

[96] L. Hong, A. S. Doumith, and B. D. Davison. Co-factorization machines: modeling user interests and predicting individual decisions in Twitter. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 557–566, New York, NY, USA, 2013.

[97] L. Hong, Z. Yang, and B. D. Davison. Incorporating participant reputation in community-driven question answering systems. *IEEE International Conference on Computational Science and Engineering*, 4:475–480, 2009.

[98] L. Hong, D. Yin, J. Guo, and B. D. Davison. Tracking trends: incorporating term volume into temporal topic models. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 484–492, New York, NY, USA, 2011.

[99] B. Hu, Y. Zhang, W. Chen, G. Wang, and Q. Yang. Characterizing search intent diversity into click models. In *WWW 2011*, pages 17–26.

[100] M. Hu, E.-P. Lim, A. Sun, H. W. Lauw, and B.-Q. Vuong. On improving Wikipedia search using article quality. In *Proceedings of the 9th Annual ACM International Workshop on Web Information and Data Management (WIDM)*, pages 145–152, New York, NY, 2007. ACM.

[101] J. Huang, M. Zhou, and D. Yang. Extracting chatbot knowledge from online discussion forums. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 423–428, Jan. 2007.

[102] I. Hutchby and R. Wooffitt. *Conversation Analysis*. Polity, 2008.

[103] T. Iwata, T. Yamada, Y. Sakurai, and N. Ueda. Online multiscale dynamic topic models. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 663–672, 2010.

[104] A. Java, X. Song, T. Finin, and B. Tseng. Why we Twitter: Wnderstanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD Workshop on Web Mining and Social Network Analysis*, pages 56–65, 2007.

[105] J. Jeon, W. B. Croft, and J. H. Lee. Finding semantically similar questions based on their answers. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 617–618, New York, NY, 2005. ACM.

[106] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 84–90, New York, NY, 2005. ACM.

[107] J. Jeon, W. B. Croft, J. H. Lee, and S. Park. A framework to predict the quality of answers with non-textual features. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 228–235, New York, NY, 2006. ACM.

[108] V. Jijkoun and M. de Rijke. Retrieving answers from frequently asked questions pages on the web. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 76–83, New York, NY, 2005. ACM.

[109] T. Joachims. Optimizing search engines using clickthrough data. In *KDD 2002*, pages 133–142.

[110] P. Jurczyk and E. Agichtein. Discovering authorities in question answer communities by using link analysis. In *CIKM '07: Proceedings of the 16th ACM Conference on Information and Knowledge Management*, pages 919–922, New York, NY, USA, 2007. ACM.

[111] N. Kawamae and R. Higashinaka. Trend detection model. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 1129–1130, New York, NY, USA, 2010. ACM.

[112] E. Khabiri, C.-F. Hsu, and J. Caverlee. Analyzing and predicting community preference of socially generated metadata: A case study on comments in the digg community. In *ICWSM 2009*, 2009.

[113] J. Kim, G. Chern, D. Feng, E. Shaw, and E. Hovy. Mining and assessing discussions on the web through speech act analysis. In *Proceedings of the Workshop on Web Content Mining with Human Language Technologies at the 5th International Semantic Web Conference*, 2006.

[114] J. Kim, E. Shaw, D. Feng, C. Beal, and E. Hovy. Modeling and assessing student activities in on-line discussions. In *Proceedings of the Workshop on Educational Data Mining at AAAI*, 2006.

[115] Y. Kim and K. Shim. Twitobi: A recommendation system for twitter using probabilistic modeling. In *ICDM*, pages 340 –349, 2011.

[116] J. Kleinberg. Bursty and hierarchical structure in streams. *Journal Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.

[117] J. Kleinberg. Temporal dynamics of on-line information streams. In *Data Stream Management: Processing High-Speed Data Streams*, 2005.

[118] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46:604–632, September 1999.

[119] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys 2011*, pages 165–172.

[120] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51:455–500, August 2009.

[121] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM TKDD*, 4:1:1–1:24, January 2010.

[122] Y. Koren and J. Sill. OrdRec: an ordinal model for predicting personalized item rating distributions. In *RecSys 2011*, pages 117–124.

[123] B. Krishnamurthy, P. Gill, and M. Arlitt. A few chirps about twitter. In *Proceedings of the 1st Workshop on Online Social Networks*, pages 19–24, 2008.

[124] N. Kushmerick, T. Lau, M. Dredze, and R. Khoussainov. Activity-centric email: a machine learning approach. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, pages 1634–1637. AAAI Press, 2006.

[125] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.

[126] S. Lacoste-Julien, F. Sha, and M. I. Jordan. DiscLDA: Discriminative learning for dimensionality reduction and classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[127] H. Lee, R. Raina, A. Teichman, and A. Y. Ng. Exponential family sparse coding with applications to self-taught learning. In *IJCAI*, pages 1113–1119, 2009.

[128] K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: social honeypots + machine learning. In *SIGIR 2010*, pages 435–442, 2010.

[129] J. Leibenluft. A librarian's worst nightmare: Yahoo! answers, where 120 million users can be wrong. *Slate Magazine*, Dec. 2007.

[130] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 497–506, 2009.

[131] C. Li, J. Weng, Q. He, Y. Yao, A. Datta, A. Sun, and B.-S. Lee. Twiner: named entity recognition in targeted twitter stream. In *Proceedings of the 35th international ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 721–730, New York, NY, USA, 2012. ACM.

[132] C. Lin, Q. Mei, J. Han, Y. Jiang, and M. Danilevsky. The joint inference of topic diffusion and evolution in social communities. In *ICDM*, pages 378 –387, dec. 2011.

[133] C. Lin, J.-M. Yang, R. Cai, X.-J. Wang, and W. Wang. Simultaneously modeling semantics and structure of threaded discussions: a sparse coding approach and its applications. In *Proceedings of the 32nd International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 131–138, New York, NY, USA, 2009.

[134] C.-J. Lin and C.-H. Cho. Question pre-processing in a QA system on internet discussion groups. In *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, 2006.

[135] C. Liu, F. Guo, and C. Faloutsos. Bbm: bayesian browsing model from petabyte-scale data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 537–546, New York, NY, USA, 2009. ACM.

[136] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

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

[138] Y. Liu, S. Li, Y. Cao, C.-Y. Lin, D. Han, and Y. Yu. Understanding and summarizing answers in community-based question answering services. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 497–504, Manchester, UK, August 2008.

[139] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[140] T. Masada, D. Fukagawa, A. Takasu, T. Hamada, Y. Shibata, and K. Oguri. Dynamic hyperparameter optimization for Bayesian topical trend analysis. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 1831–1834, 2009.

[141] A. McCallum, X. Wang, and N. Mohanty. Joint group and topic discovery from relations and text. In *Statistical Network Analysis: Models, Issues and New Directions*, volume 4503 of *Lecture Notes in Computer Science*, pages 28–44. Springer-Verlag, 2007.

[142] Q. Mei, C. Liu, H. Su, and C. Zhai. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, 2006.

[143] Q. Mei and C. Zhai. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005.

[144] T. P. Minka. Estimating a Dirichlet distribution. *Technical Report*, 2003.

[145] S. Muralidharan, L. Rasmussen, D. Patterson, and J.-H. Shin. Hope for haiti: An analysis of facebook and twitter usage during the earthquake relief efforts. *Public Relations Review*, 37(2):175 – 177, 2011.

[146] M. Naaman, J. Boase, and C.-H. Lai. Is it really about me?: message content in social awareness streams. In *CSCW 2010*, pages 189–192.

[147] R. M. Nallapati, A. Ahmed, E. P. Xing, and W. W. Cohen. Joint latent topic models for text and citations. In *KDD '08: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 542–550, 2008.

[148] R. M. Nallapati, S. Ditmore, J. D. Lafferty, and K. Ung. Multiscale topic tomography. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 520–529, 2007.

[149] A. Neumaier and T. Schneider. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Transactions on Mathematical Software*, 27:27–57, March 2001.

[150] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed algorithms for topic models. *The Journal of Machine Learning Research*, 10:1801–1828, 2009.

[151] M. Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[152] L. Nie, B. D. Davison, and X. Qi. Topical link analysis for web search. In *Proc. of the 29th Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 91–98, 2006.

[153] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[154] M. Paul. Cross-collection topic models: Automatically comparing and contrasting text. Master's thesis, UIUC, 2009.

[155] M. Paul and R. Girju. Cross-cultural analysis of blogs and forums with mixed-collection topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1408–1417. Association for Computational Linguistics, 2009.

[156] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, pages 215–224, Los Alamitos, CA, 2001. IEEE Computer Society.

[157] H.-K. Peng, J. Zhu, D. Piao, R. Yan, and Y. Zhang. Retweet modeling using conditional random fields. In *ICDM Workshops*, pages 336–343, 2011.

[158] J. Petterson, A. Smola, T. Caetano, W. Buntine, and S. Narayanamurthy. Word features for latent dirichlet allocation. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1921–1929. 2010.

[159] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceeding of the 17th International Conference on World Wide Web*, pages 91–100, 2008.

[160] O. Phelan, K. McCarthy, and B. Smyth. Using Twitter to recommend real-time topical news. In *RecSys 2009*, pages 385–388, 2009.

[161] I. Pruteanu-Malinici, L. Ren, J. Paisley, E. Wang, and L. Carin. Hierarchical Bayesian modeling of topics in time-stamped documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:996–1011, June 2010.

[162] T. Qin, T.-Y. Liu, and H. Li. A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*, 13:375–397, 2010.

[163] D. Ramage, S. Dumais, and D. Liebling. Characterizing microblogs with topic models. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, 2010.

[164] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled lda: a supervised topic model for credit attribution in multi-labeled corpora. In *EMNLP '09: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 248–256. Association for Computational Linguistics, 2009.

[165] S. Rendle. Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57:1–57:22, 2012.

[166] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *KDD 2009*, pages 727–736.

[167] S. Rendle, C. Freudenthaler, Z. Gantner, and S.-T. Lars. BPR: Bayesian personalized ranking from implicit feedback. In *UAI 2009*, pages 452–461.

[168] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM 2010*, pages 81–90.

[169] S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

[170] A. Ritter, C. Cherry, and B. Dolan. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 172–180, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[171] M. Rosen-Zvi, C. Chemudugunta, T. Griffiths, P. Smyth, and M. Steyvers. Learning author-topic models from text corpora. *ACM Transactions on Information Systems*, 28(1):1–38, 2010.

[172] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *UAI '04: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 487–494, 2004.

[173] H. Sacks, E. A. Schegloff, and G. Jefferson. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735, 1974.

[174] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 377–386, 2006.

[175] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML 2008*, pages 880–887.

[176] J. Seo, W. B. Croft, and D. A. Smith. Online community search using thread structure. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 1907–1910, New York, NY, USA, 2009. ACM.

[177] D. Shamma, L. Kennedy, and E. Churchill. Tweetgeist: Can the Twitter timeline reveal the structure of broadcast events? In *Proceeding of the 2010 ACM Conference on Computer Supported Cooperative Work (CSCW)*, 2010.

[178] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 1025–1030, Washington, DC, USA, 2010. IEEE Computer Society.

[179] L. Shrestha and K. McKeown. Detection of question-answer pairs in email conversations. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, page 889, Morristown, NJ, 2004. Association for Computational Linguistics.

[180] A. P. Singh. *Efficient Matrix Models for Relational Learning*. PhD thesis, Machine Learning Department, Carnegie Mellon University, Oct. 2009.

[181] A. P. Singh and G. J. Gordon. A unified view of matrix factorization models. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pages 358–373, Berlin, Heidelberg, 2008. Springer-Verlag.

[182] S. Sizov. Geofolk: latent spatial semantics in web 2.0 social media. In *Proceedings of WSDM 2010*, pages 281–290, New York, NY, USA. ACM.

[183] A. Smola and S. Narayanamurthy. An architecture for parallel topic models. *Proceedings of the VLDB Endowment*, 3(1-2):703–710, Sept. 2010.

[184] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, August 2004.

[185] Y.-I. Song, C.-Y. Lin, Y. Cao, and H.-C. Rim. Question utility: A novel static ranking of question search. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, July 2008.

[186] B. Suh, L. Hong, P. Pirolli, and E. H. Chi. Want to be retweeted? large scale analytics on factors impacting retweet in Twitter network. In *SocialCom 2010*, pages 177–184, 2010.

[187] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers on large online QA collections. In *46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, 2008.

[188] K. Toutanova and C. D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 63–70. Association for Computational Linguistics, 2000.

[189] J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[190] N. Usunier, D. Buffoni, and P. Gallinari. Ranking with ordered weighted pairwise classification. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 1057–1064, New York, NY, USA, 2009.

[191] I. Uysal and W. B. Croft. User oriented tweet ranking: a filtering approach to microblogs. In *CIKM*, pages 2261–2264, 2011.

[192] E. M. Voorhees. The TREC question answering track. *Nat. Lang. Eng.*, 7(4):361–378, 2001.

[193] H. M. Wallach. Topic modeling: beyond bag-of-words. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, 2006.

[194] H. M. Wallach, D. Mimno, and A. McCallum. Rethinking LDA: Why priors matter. In *Proceedings of NIPS*, 2009.

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

[196] C. Wang, D. M. Blei, and D. Heckerman. Continuous time dynamic topic models. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 579–586, 2008.

[197] C. Wang, J. W. Paisley, and D. M. Blei. Online variational inference for the hierarchical dirichlet process. *Journal of Machine Learning Research - Proceedings Track*, 15:752–760, 2011.

[198] C. Wang, J. Wang, X. Xie, and W.-Y. Ma. Mining geographic knowledge using location aware topic model. In *Proceedings of the 4th ACM workshop on Geographical information retrieval*, GIR '07, pages 65–70, New York, NY, USA, 2007. ACM.

[199] K. Wang, T. Walker, and Z. Zheng. Pskip: estimating relevance ranking quality from web search clickthrough data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1355–1364, New York, NY, USA, 2009. ACM.

[200] L. Wang and D. W. Oard. Context-based message expansion for disentanglement of interleaved text conversations. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL/HLT)*, pages 200–208, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[201] X. Wang and A. McCallum. Topics over time: A non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 424–433, 2006.

[202] Y.-C. Wang, M. Joshi, W. W. Cohen, and C. P. Rosé. Recovering implicit thread structure in newsgroup style conversations. In *Proceedings of the 2nd International Conference on Weblogs and Social Media (ICWSM)*, 2008.

[203] Y.-C. Wang, M. Joshi, and C. P. Rosé. A feature based approach to leveraging context for classifying newsgroup style discussion segments. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions (ACL)*, pages 73–76, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

[204] X. Wei and W. B. Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185, 2006.

[205] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. CofiRank - maximum margin matrix factorization for collaborative ranking. In *NIPS*, pages 1593–1600, 2007.

[206] M. Weimer, A. Karatzoglou, and A. Smola. Adaptive collaborative filtering. In *RecSys*, pages 275–282, 2008.

[207] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *WSDM '10: Proceedings of the Third ACM International Conference on Web Search and Data Mining*, pages 261–270, 2010.

[208] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning*, 81(1):21–35, Oct. 2010.

[209] J. Weston, C. Wang, R. Weiss, and A. Berenzweig. Latent collaborative retrieval. In *ICML*, 2012.

[210] B. P. Wing and J. Baldridge. Simple supervised document geolocation with geodesic grids. In *Proceedings of ACL 2011*, pages 955–964, Stroudsburg, PA, USA. Association for Computational Linguistics.

[211] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of SIAM Data Mining*, 2010.

[212] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–482, 2008.

[213] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 537–546, New York, NY, USA, 2011. ACM.

[214] S.-H. Yang, B. Long, A. J. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 295–304, New York, NY, USA, 2011. ACM.

[215] Z. Yang, J. Guo, K. Cai, J. Tang, J. Li, L. Zhang, and Z. Su. Understanding retweeting behaviors in social networks. In *CIKM*, pages 1633–1636, 2010.

[216] L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 937–946, 2009.

[217] S. Ye and F. Wu. Measuring message propagation and social influence on Twitter.com. In *Proceedings of The 2nd International Conference on Social Informatics*, pages 216–231, 2010.

[218] W.-T. Yih and C. Meek. Improving similarity measures for short segments of text. In *AAAI'07: Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 1489–1494, 2007.

[219] Z. Yin, L. Cao, J. Han, C. Zhai, and T. Huang. Geographical topic discovery and comparison. In *Proceedings of WWW 2011*, pages 247–256, New York, NY, USA. ACM.

[220] G.-X. Yuan, C.-H. Ho, and C.-J. Lin. Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584 –2603, Sept. 2012.

[221] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR 2007*, pages 271–278.

[222] C. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 743–748, 2004.

[223] H. Zhang, C. L. Giles, H. C. Foley, and J. Yen. Probabilistic community discovery using hierarchical latent gaussian mixture model. In *AAAI'07: Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 663–668, 2007.

[224] J. Zhang, Y. Song, C. Zhang, and S. Liu. Evolutionary hierarchical Dirichlet processes for multiple correlated time-varying corpora. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1079–1088, 2010.

[225] Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *KDD 2011*, pages 1388–1396.

[226] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing Twitter and traditional media using topic models. In *ECIR*, pages 338–349, 2011.

[227] D. Zhou, S. A. Orshanskiy, H. Zha, and C. L. Giles. Co-ranking authors and documents in a heterogeneous network. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 739–744, Washington, DC, USA, 2007. IEEE Computer Society.

[228] L. Zhou and E. Hovy. Digesting virtual "geek" culture: the summarization of technical internet relay chats. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 298–305, 2005.

[229] J. Zhu, A. Ahmed, and E. P. Xing. MedLDA: maximum margin supervised topic models for regression and classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1257–1264, New York, NY, USA, 2009. ACM.

[230] J. Zhu and E. P. Xing. Sparse topical coding. In *UAI*, pages 831–838, 2011.

[231] X. Zhu, D. M. Blei, and J. Lafferty. TagLDA: Bringing document structure knowledge into topic models. Technical Report TR-1553, University of Wisconsin, Madison, 2006.

[232] M. Zinkevich, A. Smola, M. Weimer, , and L. Li:. Parallelized stochastic gradient descent. In *NIPS 2010*, pages 1–9.

Vita

1985	Born in Chengdu, Sichuan Province, China.
2003	Graduated from the Chengdu No. 7 high school, Chengdu, China.
2007	B.S. in Computer Science, Beijing University of Chemical Technology, China.
2008	Summer internship at Software Consulting Services, LLC at Nazareth, PA.
2010	M.S. in Computer Science, Lehigh University.
2010	Summer internship at Yahoo! Labs, Sunnyvale, CA.
2011	Summer internship at Yahoo! Labs, Sunnyvale, CA.
2011	Fall internship at LinkedIn Corp., Mountain View, CA.
2007 - 2013	Graduate study in Department of Computer Science and Engineering, Lehigh University.
2013	Joined Yahoo! Labs as a Research Scientist.

2013	A. Ahmed, **L. Hong** and A. Smola. **Hierarchical Geographical Modeling of User locations from Social Media Posts**. In **WWW 2013**. [Full Paper / Acceptance Rate: 15%]
2013	**L. Hong**, A. Doumith and B. D. Davison. **Co-Factorization Machines: Modeling User Interests and Predicting Individual Decisions in Twitter**. In **WSDM 2013**. [Full paper / Acceptance Rate: 19%]

2012 **L. Hong**, A. Doumith and B. D. Davison. **Personalized Retweet Prediction in Twitter**. In the 4th Workshop on Information in Networks (WIN 2012).

2012 **L. Hong**, R. Bekkerman, J. Adler and B. D. Davison. **Learning to Rank in Social Update Streams**. In **SIGIR 2012**. [Full Paper / Acceptance Rate: 20%]

2012 **L. Hong**, A. Ahmed, S. Gurumurthy, A. Smola and K. Tsioutsiouliklis. **Discovering Geographical Topics in the Twitter Stream**. In **WWW 2012**. [Full Paper / Acceptance Rate: 12%]

2011 D. Yin, **L. Hong** and B. D. Davison. **Structural Link Analysis and Prediction in Microblogs**. In **CIKM 2011**. [Short Paper / Acceptance Rate: 35%]

2011 **L. Hong**, D. Yin, J. Guo and B. D. Davison. **Tracking Trends: Incorporate Volume into Temporal Topic Models**. In **KDD 2011**. [Full Paper / Oral Presentation / Acceptance Rate: 17.5%]

2011 **L. Hong**, B. Dom, S. Gurumurthy and K. Tsioutsiouliklis. **A Time-Dependent Topic Model for Multiple Text Streams**. In **KDD 2011**. [Full Paper / Poster Presentation / Acceptance Rate: 17.5%]

2011 D. Yin, **L. Hong**, Z. Xue and Brian D. Davison. **Temporal Dynamics of User Interests in Tagging Systems**. In **AAAI 2011**. [Full Paper/Acceptance Rate: 24%]

2011 D. Yin, **L. Hong**, X. Xiong and B. D. Davison. **Link Formation Analysis in MicroBlogs**. In (**SIGIR 2011**). [Poster/Acceptance Rate: 32%]

2011 **L. Hong**, O. Dan and B. D. Davison. **Predicting Popular Messages in Twitter**. In (**WWW 2011**). [Poster] **Best Poster Award**.

2011 D. Yin, **L. Hong** and B. D. Davison. **Exploiting Session-like Behaviors in Tag Prediction**. In **WWW 2011**. [Poster]

2010 **L. Hong** and B. D. Davison. **Empirical Study of Topic Modeling in Twitter**. In SOMA 2010.

2010 D. Yin, Z. Xue, **L. Hong** and B. D. Davison. **A Probabilistic Model for Personalized Tag Prediction**. In **KDD 2010**. [Full Paper/Acceptance Rate: 17%]

2010 Z. Yang, **L. Hong** and B. D. Davison. **Topic-driven Multi-type Citation Network Analysis**. In RIAO 2010. [Full Paper/Acceptance Rate: 19%]

2010 **L. Hong** and B. D. Davison. **Wanted: A Unified Model for Search in Social Media** [Position Paper]. In SSM 2010. [Panel]

2009 J. Wang, **L. Hong** and B. D. Davison. **Tag Recommendation Using Keywords and Association Rules**. In 2009 ECML PKDD Discovery Challenge Workshop. [Poster]

2009 **L. Hong**, Z. Yang and B. D. Davison. **Incorporating Participant Reputation in Community-driven Question Answering Systems**. In SIN09. [Full Paper/Acceptance Rate: 20%]

2009 **L. Hong** and B. D. Davison. **A Classification-Based Approach to Question Answering in Discussion Boards**. In **SIGIR 2009**. [Full Paper/Acceptance Rate: 16%]

2009 D. Yin, Z. Xue, **L. Hong**, B. D. Davison, A. Kontostathis and L. Edwards. **Detection of Harassment on Web 2.0**. In Content Analysis in Web 2.0 Workshop. [Full Paper]