CONTEXTUAL TEXT MINING

BY

QIAOZHU MEI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2009

Urbana, Illinois

Doctoral Committee:

Associate Professor ChengXiang Zhai, Chair
Professor Jiawei Han
Professor Bruce R. Schatz
Professor Kenneth W. Church, Johns Hopkins University

Abstract

With the dramatic growth of text information, there is an increasing need for powerful text mining systems that can automatically discover useful knowledge from text. Text is generally associated with all kinds of contextual information. Those contexts can be explicit, such as the time and the location where a blog article is written, and the author(s) of a biomedical publication, or implicit, such as the positive or negative sentiment that an author had when she wrote a product review; there may also be complex context such as the social network of the authors. Many applications require analysis of topic patterns over different contexts. For instance, analysis of search logs in the context of the user can reveal how we can improve the quality of a search engine by optimizing the search results according to particular users; analysis of customer reviews in the context of positive and negative sentiments can help the user summarize public opinions about a product; analysis of blogs or scientific publications in the context of a social network can facilitate discovery of more meaningful topical communities. Since context information significantly affects the choices of topics and language made by authors, in general, it is very important to incorporate it into analyzing and mining text data. In general, modeling the context in text, discovering contextual patterns of language units and topics from text, a general task which we refer to as Contextual Text Mining, has widespread applications in text mining.

In this thesis, we provide a novel and systematic study of contextual text mining, which is a new paradigm of text mining treating context information as the "first-class citizen." We formally define the problem of contextual text mining and its basic tasks, and propose a general framework for contextual text mining based on generative modeling of text. This conceptual framework provides general guidance on text mining problems with context information and can be instantiated into many real tasks, including the general problem of contextual topic analysis. We formally present a functional framework for contextual topic analysis, with a general contextual topic model and its various versions, which can effectively solve the text mining problems in a lot of real world applications.

We further introduce general components of contextual topic analysis, by adding priors to contextual topic models to incorporate prior knowledge, regularizing contextual topic models with dependency structure of

context, and postprocessing contextual patterns to extract refined patterns. The refinements on the general contextual topic model naturally lead to a variety of probabilistic models which incorporate different types of context and various assumptions and constraints. These special versions of the contextual topic model are proved effective in a variety of real applications involving topics and explicit contexts, implicit contexts, and complex contexts.

We then introduce a postprocessing procedure for contextual patterns, by generating meaningful labels for multinomial context models. This method provides a general way to interpret text mining results for real users.

By applying contextual text mining in the "context" of other text information management tasks, including ad hoc text retrieval and web search, we further prove the effectiveness of contextual text mining techniques in a quantitative way with large scale datasets.

The framework of contextual text mining not only unifies many explorations of text analysis with context information, but also opens up many new possibilities for future research directions in text mining.

To Yanhua;

to my parents.

Acknowledgments

First of all, I wish to express my deepest gratitude to my advisor, Dr. ChengXiang Zhai. Without Cheng's continuous guidance and help throughout my PhD study, this thesis wouldn't have been completed. Cheng showed me how to identify and formulate interesting research problems, how to develop elegant and practical solutions, and how to evaluate a technique with comprehensive experiments. Cheng is the one who has helped me grow from an infant into someone who can walk by himself in research. In the past five years, I have learned so much from Cheng - the great passion, the broad vision, the high standard, and the serious attitude in research. From the very beginning, he has been treating me as a peer and a friend, and I am deeply grateful to him not only for his constructive technical guidance, but also for his continuous encouragement and the freedom he has given me in my research. Working with him has been a great joy to me. Cheng's guidance and personality has deeply influenced both my research philosophy and my career choice. I hope I will be a great advisor like him for my own students.

I wish to express my thanks to the other members of my committee, Dr. Ken Church, Dr. Jiawei Han, and Dr. Bruce Schatz for their critical suggestions about the thesis.

I have many reasons to thank Dr. Ken Church. I worked with Ken for two summers at Microsoft Research. Ken is much more than a mentor to me. His unique perspective and vision of research has brought me to a realm I've never been to. His special sense in large scale data mining and Shannon's entropy has a large impact on my research. I am specifically thankful for his tremendous help on this thesis, especially on the problem of personalized search, and his continuous support on my study and career.

Dr. Jiawei Han has given me so many suggestions and support all through my study in UIUC. In his course I built up my data mining knowledge basis, and from the discussion and collaboration with him I learned how to explore in the mysterious frontier of data mining.

I am deeply grateful to Dr. Bruce R. Schatz. I worked as a research assistant of Bruce for two years in the BeeSpace project, through which I acquired the skills to make theoretical methods practical in real world applications. Bruce has given me a lot of encouragements and great advice. His vision in bridging state-of-the-art techniques with real applications has influenced me a lot.

I would like to express my thanks to my mentors at Yahoo! Research, Dr. Andrew Tomkins and Dr. Ravi Kumar, and my colleague Dr. Dengyong Zhou at Microsoft. With them I had many valuable discussions and from them I learned a lot of methodology and attitude of doing research. I'd like to give my special thanks to Dr. Yi Zhang, for her continuous encouragement and help during my study.

In all the past five years, I have received help from many collaborators, colleagues, and friends at UIUC. I would like to express my thanks to the members of the TIMan Group for the their valuable discussions and help: Tao Tao, Hui Fang, Xuehua Shen, Jing Jiang, Azadeh Shakery, Bin Tan, Xu Ling, Xuanhui Wang, Xin He, Younhee Ko, Yue Lu, Maryam Karimzadehgan, Alex Kotov, Duo Zhang, Yuanhua Lv, and V.G.Vinod Vydiswaran. I would also like to thank many of my fiends in the DAIS Group and the computer science department, especially Xifeng Yan, Dong Xin, Chao Liu, Deng Cai, Tianyi Wu, Tao Cheng, Zhijun Yin, Rui Li, Yunliang Jiang, and Bran Chee. We had many fruitful discussions and spent a joyful time together in UIUC which I will never forget.

I want to express my great thanks to Yahoo! Inc. and Roy J. Carver Charitable Trust for granting me the Yahoo! PhD Fellowship and the Roy J. Carver Fellowship to support my graduate study.

Finally, I would like to thank my dear wife Yanhua and my parents for their love and strong support for my study and career. Without their encouragement and help I couldn't have reached this far. This thesis is dedicated to them.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the novel "Gulliver's Travels," when Gulliver was on his first voyage to the land Lilliput, he appeared as a giant who could easily dominate a battle; when he was on his second voyage to the land Brobdingnag, however, he appeared as small as a rat who could only be exhibited as a curiosity. If Don Quijote were born in the time of King Arthur, he might have made a real famous knight. However, at his own time when the era of chivalry was long gone, all his dreams turned into nothing but laughters of others. The conversion and mismatch of context account for the dramatic difference in their experiences.

The impact of context not only exists in fictional stories, but also in our everyday activities of reading and digesting text information. In our daily lives, we observe that today's newspaper talks about totally different topics from *one week ago*. We find out that a *Republican senator* has opposite opinion to a *Democratic senator* on the same event. We read a report about "football" on *BBC*, and realize that it is all about soccer instead of American football. A researcher summarizes topics in literature, and tries to identify the topics that are getting hot over *recent years*. A user types a query "moving service" in the search box, bypasses the top-ranked results, and selects a moving company on the second page which is in her *city*. A *computer scientist* searches for "msr" and clicks on Microsoft research instead of Mountain Safety Research. A tourist reads restaurant reviews, and visits a restaurant with *positive* comments. A little boy tries to guess the meaning of an unknown word by looking at the words *adjacent to* that word. In all these scenarios, we rely much on the understanding of *context* in the process of understanding a piece of *text*.

Text data is associated with rich context information, which refers to the situations in which the text was originally produced. Some of the situations can be easily inferred from the metadata, such as the time at which an article was written, the source at which an article was published, the author, the audience, and the sentence preceding the current sentence. Some of the situations are implicit and thus require more effort to infer them from the data, such as the sentiment of the author when she was writing a review and the topics she wanted to elaborate in a research paper. Some contexts are global, where a whole document belongs to the same context; some contexts appear at a finer granularity, such as a sentence, a phrase, or a window of adjacent words. Various contexts can be dependent on each other and thus form a complex system, such as

a social network of authors.

There are different taxonomies of context from different disciplines. For example, the linguistic context, or the verbal context is commonly used in linguistics which refers to the local surrounding text of a linguistic unit that is useful for inferring the meaning of the linguistic unit [168]. Social context, on the other hand, is a key concept used in sociology which refers to the social variables that influence the use of language of a social identity (e.g., an author). A more general notion of context is known as the situational context, or context of situation, which is first proposed by the Polish anthropologist Bronislaw Malinowski and then formalized with linguistic theory by J. R. Firth [43, 44]. It is concerned with the evaluable conditions or situations in which the text content is produced, including the situations that are either environmental or personal.

Figure 1.1: Rich context information in Twitter.

Figure 1.1 is a random snapshot from social microblogging site Twitter[1]. It is interesting to see how many types of context information exist in such a short piece of text. From the metadata, we can easily observe who wrote this article (*authorship*), where the author lived (*location*), what his occupation was (*social context*), when the article was written (*time*), from which *platform* the article was submitted, and in which *language* the article was written, etc. When reading through the text, we can further observe contexts that are implicit - the *sentiment* that the author implied in the article, the *source* of a statement

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

in the article, and the *impact* of the article (based on the followers and comments). We can even observe a complex system of contexts - a *social network* including the author, his followers, and the users he follows. All such contextual information is valuable for analyzing the content of the Tweets (i.e., posts on Twitter), understanding why the author wrote about these topics, inferring the meaning of the text (e.g., the metaphor carried by the word "well"), and revealing other interesting topic patterns.

Understanding the effects of context is becoming even more important along with the *information overload* [177] in recent years. Various types of text information, such as web pages, news articles, scientific literature, emails, blogs, and instant messages, are continuously produced everywhere. According to Ramakrishnan and Tomkins [145], everyday there are 3-4 Gigabytes of published content, up to 2 Gigabytes of professional web content, 8-10 Gigabytes of user-generated content, and as large as 3 Terabytes of private text content produced. The scale of text information is no longer manageable by individuals. Instead, there is an urgent need for powerful text information management systems to combat the information overload, to facilitate the filtering, storage, and retrieval of relevant text information. There is a notably increasing need for going beyond finding text information - by discovering novel knowledge from the text data, also known as *text mining*. Now not only we ourselves should understand the importance of context, but also the text mining tools.

Indeed, in many real world applications of text mining, we can see that various context information can serve as an important guidance for understanding, analyzing, and mining the text data. For example, temporal patterns of book discussions in blogs (e.g., spikes of discussions) have shown to be useful to predict books sales [52]. Author-topic patterns can facilitate the finding of experts and the understanding of the research communities [162]. Analyzing contextual patterns in search logs can help a search engine company to better serve its customers by re-organizing the search results according to the contexts of a new query [108]. Analyzing the bursts and decays of topics in scientific literature would enable researchers to better organize, summarize, and digest the historical literature and to discover and envision new research trends [72]. Analyzing the sentiments in customer reviews is helpful in summarizing public opinions about products and social events [110].

Unfortunately, although researchers have long revealed that context of text is important, the exploration of context in text mining doesn't reflect this importance. In most existing text information management systems, the importance of context is neglected. For example, search engines are by far the most useful tools to help users find and access text information. However, none of the major search engines returns a webpage about "information retrieval" on the first page when the query "IR" is sent by a researcher from the SIGIR conference. University of Michigan football is not among the top results when one searches for

"wolverine" from Ann Arbor, Michigan. The results of the query "Wii fit review" don't distinguish positive reviews from negative reviews. If we know who the user is, where she is, and what task she is carrying on, we should be able to provide much better service for her by adapting the results to the context.

Among the particular types of context, linguistic context is utilized in natural language processing (NLP) to extend the feature space for supervised learning tasks such as tagging and parsing [30, 18], entity extraction [131], and semantic role labeling [48, 91]. It is also used directly to solve problems such as word sense disambiguation [136] and word clustering [92, 93], in a way that the meaning of language units are compared through the comparison of their local linguistic context. The exploration of the more general context, context of situation, is quite limited in existing text mining work. In this thesis, we focus on this more general notion of context, context of situation, in the "context" of text mining.

A common approach for text mining relies on generative models of text, the key idea of which is to construct a probabilistic model which explains how the observed text data is generated. Such a probabilistic model of text, also known as a language model, is then used to characterize the patterns (e.g., topics) in text or to facilitate tasks like retrieval [138, 182], classification, clustering, and summarization of text. In the traditional exploration of text mining, people assume that there is a unitary, homogenous language model for the collection of text data, thus the contextual variation is neglected. Early exploration of context in text mining dates back to the 60s, where Mosteller and Wallace used statistical methodology to solve one of the most famous questions in America's history: the disputed authorship of the Federalist Papers [125]. The basic assumption was that text written by different authors have different characteristics (thereby each author makes a different context). Later on, the construction of the Brown Corpus [45] revealed important statistics showing that the content of text (e.g., the word rate) differs from document to document and from genre to genre. Motivated by this observation, [27] makes a visionary claim that a good model for a text collection should capture the mixture of the heterogeneous contextual factors. They concluded that a mixture Poisson distribution fits text data better than standard Poisson models. However, this early exploration halted at collection-level modeling of text data as a black box. They did not drill down to understand what the contexts are and how to model particular contexts. How to define and characterize the contexts in text, how to capture the effect of these contexts, and how to utilize context information to solve various tasks of text mining still remained mysterious.

A notable advancement of modeling the mixture of contexts is the recent exploration on probabilistic topic models (e.g., [61, 10]). The basic assumption is that text data usually covers a set of multiple topics, and can thus be modeled by a mixture of probabilistic models, each of which corresponds to a topic. On the other hand, one can characterize a topic based on the corresponding component model. This exploration

advanced beyond the Poisson mixture in a way that we can now characterize concrete topics in text, where each topic is an implicit context. However, the topic models lack the ability to model the much richer context information other than topics. We also don't know how to integrate our prior understanding about contexts into a topic model and how the topics themselves are affected by various contexts.

Some recent work tries to integrate specific context information into topic models, such as authorship [162] and time [171]. These specific types of context information are integrated in an ad hoc way. The proposed models are difficult to be extended to handle other context information.

In general, the major barrier in utilizing context information in text mining is that there is no general way of defining context, modeling context, comparing content across contexts, and discovering contextual patterns from text. Without such a general framework, text mining tasks with context information involved have to be solved in a case-by-case manner.

In this thesis, we provide a formal study of modeling context information in text mining. We propose a general problem, contextual text mining, which is a new paradigm of text mining that treats context information as the "first class citizen." We provide a general conceptual framework of contextual text mining, in which we formally define a context as an evaluable situation in which the text data is generated, define contextual patterns as a variety of conditional distributions of language units and contexts, and define the core tasks of contextual text mining as the construction of a contextualized language model of text and the extraction of contextual patterns. The modeling of context is then realized by constructing a mixture of contextualized language models of text. This conceptual framework is very general, which unifies many existing explorations of contextual text mining and many existing language models as special cases.

Based on this general conceptual framework, we introduce a functional framework of contextual text mining called contextual topic analysis, the main idea is to integrate context variables into a probabilistic topic model in order to construct a contextual topic model (i.e., the contextual probabilistic latent semantic analysis (CPLSA)), such that both the content variation of topics and the strength variation of topics across different contexts are captured by the conditional distributions of topics and contexts. This functional framework is instantiated from the conceptual framework, and still general enough to cover many different contexts and different special cases. On the other hand, the contextual topic model and its special versions are fully functional that can support a wide range of real applications.

In some scenarios, we have rich prior knowledge about what contexts are useful, and what the model of those contexts should look like. In other scenarios we don't have any guidance on the contexts. In our framework, we propose a general component to integrate prior knowledge of contexts into the contextual topic models, so that if we have prior knowledge about a context, we can integrate such knowledge into the

model as a guidance; if we don't have such prior knowledge, the model will fully listen to the data.

Distinct from existing work which considers different contexts as independent, in the proposed framework, we provide a general component to model the dependency among various contexts as a complex system. The modeling of context dependency is realized with a graph-based regularization component to the contextual probabilistic models, with which we can flexibly incorporate important constraints. To reveal the dependency of contexts, we also present a postprocessing method for contextual topic models by constructing an evolutionary theme graph.

A limitation of topic modeling, or language modeling of text in general is that the probabilistic models can not be easily understood by the real users. We present a postprocessing component of contextual text mining to interpret the characteristics of various contexts by the automatic generation of meaningful labels for the corresponding contextual models.

Both the conceptual framework of contextual text mining and the functional framework of contextual topic analysis are quite general and cover many specific models as special cases. Through the framework of contextual topic analysis, we can model heterogeneous contexts in a unified way. We can characterize each context with the contextualized topic component corresponding to this context and also extract various types of contextual patterns by comparing the conditional distributions involving topics and various contexts. By instantiating the contextual topic model, we can effectively model explicit contexts such as time, location, and authorship, implicit contexts such as sentiments, as well as complex contexts such as social networks. These instantiations naturally lead to the solution for a wide range of real applications of contextual text mining. In this thesis, we present how to use specific instantiations of the framework to solve real world text mining problems such as the detection of evolutionary theme patterns, the detection of spatiotemporal topic patterns, event-impact analysis, personalized search, faceted opinion summarization, the extraction of topical communities from scientific literature, and smoothing language models in information retrieval.

The thesis provides a novel and general perspective of utilizing context information in text mining. The proposed frameworks not only unify existing models with particular types of context in text mining, but also provide a way to design solutions for new contextual text mining tasks with other interesting context information involved. The functional framework of contextual topic analysis provides a general way to guide probabilistic models with prior knowledge of users, thus enables the interaction between the user and the text mining process. It also bridges the gap between probabilistic modeling of text and graph-based regularization, through which various constraints and objectives can be easily added into the text mining process. Furthermore, this thesis provides a novel method to generate meaningful labels to interpret topic models in the text mining results. We also introduce how to utilize the contextual patterns extracted in

contextual text mining to facilitate other text information management tasks including ad hoc retrieval and web search. The framework opens up many possibilities for developing principled approaches to mining text data with context information, by designing different instantiations of the general contextual probabilistic model, designing interesting priors for contexts, and designing regularization components with reflects various constraints and assumptions about the dependency of contexts.

The rest of this thesis is organized as follows. We begin with a brief review of existing language models in text mining in Chapter 2. In Chapter 3, we formally define the problem of contextual text mining, discuss the basic principles of contextual text modeling, and introduce a very general conceptual framework of contextual text mining. To bridge the general conceptual framework with real world applications, in Chapter 4, we propose a functional framework of contextual text mining called contextual topic analysis, a general contextual topic model called CPLSA, and some special versions of CPLSA under certain constraints.

The conceptual framework of contextual text mining and the functional framework of contextual topic analysis serve as the theoretical guideline of any contextual text mining task. We then illustrate the power of the general framework and the CPLSA model with real world applications and different types of contexts. In Chapter 5, we show the performance of CPLSA and its special versions with explicit contexts besides topics. In Chapter 6, we illustrate the power of contextual topic analysis on implicit contexts (e.g., sentiments) besides topics, with a special version of CPLSA called Topic-Sentiment Mixture. We also elaborate the component of adding priors to contextual topic models in details. In Chapter 7, we further introduce how to instantiate the framework of contextual topic analysis to model complex context, such as social networks. We elaborate how to regularize a contextual topic model with a graph-based regularizer to incorporate the dependency structure of contexts.

After this, two ways of postprocessing the contextual patterns extracted in the mining process are introduced in Chapter 8 and Chapter 9. We then introduce two particular applications of contextual text mining to facilitate other text information management tasks - smoothing language models in text retrieval (Chapter 10) and providing personalization in web search (Chapter 11). These two applications not only illustrate the power of contextual text mining in a broader scope, but also provide quantitative evaluation of the effectiveness of the contextual text mining techniques with large scale data sets.

We finally present our conclusions and discuss future directions in Chapter 12.

Chapter 2

A Review of Language Models in Text Mining

Text mining is concerned with the discovery of novel knowledge from text data. A common approach to text mining is through the modeling of the generative process of text data, or the generative modeling of text. The basic idea is to construct a probabilistic model which explains how the language units in text are generated. Such a probabilistic model is also known as a *statistic language model*, or simply a *language model*. A typical text mining system first learns such a probabilistic model based on the observations of the text data, and then utilizes the probabilistic model learnt to accomplish particular tasks, such as text retrieval, text classification, text clustering, and text summarization. Alternative to generative models in text mining are discriminative models.

Different generative models of text differ in their assumptions about the generative process of text. In this thesis, we follow the line of work on generative modeling of text. Context information is incorporated into the generative process of text as dependent variables in the language models. We begin with a brief review of some commonly used language models in text mining.

2.1 Corpus Language Models

The simplest language model assumes that the text corpus is generated word by word from a unified distribution of words. The two most common distributions used in literature are the Poisson distribution and multinomial distribution. Generally, let us use w to denote a word token in the text collection, $\mathcal{M}$ to denote the language model, and $c(w)$ to denote the number of occurrences of word w in a piece of text (e.g., a document), a Poisson language model [35, 110] scan be written as

$$P(c(w) = k|\mathcal{M}) = \frac{e^{-\lambda}\lambda^k}{k!}, \tag{2.1}$$

where λ is a model parameter and $k!$ is the factorial of the number of occurrences of the word w.

Different from a Poisson distribution, a multinomial distribution models [183, 35] each individual ap-

pearance of a word instead of the number of occurrences. We have

$$P(c(w) = k) \propto P(w|\mathcal{M})^k, \tag{2.2}$$

where the model parameters are $P(w|\mathcal{M})$ (for every unique word w), usually abbreviated as θ.

There are two important assumptions in a corpus language model in text mining. First, the generation of different words in a piece of text are independent with each other. This is also known as the "bag-of-words" assumption, where the structure of the piece of text is ignored. In the context of speech recognition and machine translation, this assumption is relaxed and corpus language models are represented by "ngram" models such that the generation of a word depends on the previous word(s) [15, 63]. In the text mining "context," however, this simplification assumption is a commonly accepted even though it is too strong. Another assumption is that all the words in the corpus are generated based on the same model. We can see that this assumption brings in apparent limitations to text mining where different documents are not distinguished, not to mention the contexts.

2.2 Document Language Models

The need of text retrieval has triggered the emergence of document language models, a probabilistic model of text at document level. The basic idea is that there is a different language model for each document in the corpus, which explains the generation of words in that document. People usually assume that the document language models are the same type of distribution (e.g., multinomial distribution), but the model parameters vary in different documents. The earliest document language model used in information retrieval is introduced by Ponte and Croft [138], where each document is generated from a multi-variant Bernoulli distribution. The most commonly used distribution for a document language model is multinomial [176, 83, 183]. Using $\mathcal{D}$ to denote the document collection, θ_d to denote a document language model, and $c(w, d)$ to denote the number of occurrences of the word w in document d, the likelihood of a document is given by

$$P(\mathcal{D}|\mathcal{M}) \propto \prod_{d \in \mathcal{D}} \prod_{w \in d} P(w|\theta_d)^{c(w,d)}. \tag{2.3}$$

Some recent work also introduced document language models based on Poisson distribution [109]. In this thesis, we follow the multinomial assumption and model the individual appearances of each word in text using the bag-of-word representation of documents.

The parameters in these models are estimated based on the data observed. A general approach to

the parameter estimation of language models is maximum likelihood. The basic idea is to find the set of parameters that maximize the likelihood of the data collection to be generated from the language model. A good review of document language models can be found in [35].

2.3 Mixture Language Models

The assumption that every word in a corpus/a document is generated from the same language model is apparently too strong. There are many factors that affect the generation of words, such as genres, authors, time, and topics. Indeed, in the early work of Mosteller and Wallace [125], they concluded that text written by different authors differs from each other. Later on, the construction of the Brown Corpus [45] revealed important statistics showing that the content of text (e.g., the word rate) differs from document to document and from genre to genre. Motivated by this observation, [27] makes a visionary claim that a good model for a text collection should capture the mixture of the heterogeneous contextual factors. They concludes that a mixture Poisson distribution fits text data better than standard Poisson models. A variety of mixture Poisson models are proposed, including Two-poisson [11, 147], Katz's K-mixture [27], and negative binomial [125]. The basic idea of a mixture language model like these is that a word can be generated by flexibly selecting from one of the mixture component models, and thus leads to a new likelihood function:

$$P(x|\mathcal{M}) = \int \phi(\theta) P(x|\theta) d\theta, \tag{2.4}$$

where x can either be the appearance of word w (as in multinomial) or the number of occurrences of the word w, $c(w)$ (as in Poisson). Every unitary distribution θ is a mixture component in this mixture language model and $\phi(\theta)$ is the distribution based on which θ is selected. So $\int \phi(\theta) = 1$. There is a flexibility in selecting the number of mixture components. In models like Two-Poisson, there is a finite number of component Poisson distributions, so $\phi(\theta)$ is positive for a finite number of possible θs and zero for all others (thus called a finite mixture model) [103]. In some other cases, there are infinite number of mixture components.

Mixture language models are widely applied in text retrieval. The variation of the Two-Poisson model leads to the popular BM25 (Okapi) retrieval function, one of the most robust and effective retrieval functions [147, 148]. In the context of language modeling based retrieval, a mixture of multinomial distributions is commonly used to construct a new query model with the query language model and the feedback model [181], or to integrate a document language model with a cluster language model [97]. Zhai and Lafferty uses a 2-mixture of multinomial distributions to filter the background noise from a pseudo-feedback model [181]. Using a mixture component of the background model to absorb the noise also appears in [184, 113, 111]. In

these models, $\phi(\theta)$ are global variables in the collection, which are usually pre-specified instead of estimated.

These mixture language models utilized the intuition that there are latent contexts in text. However, they did not drill down to understand what the contexts are, and how to model particular contexts.

2.4 Probabilistic Topic Models

Recently, there is a new family of generative models of text emerging in literature with the focus on the modeling of latent topics in text. These models are known as *probabilistic topic models*, also called *statistical topic models*, or simply *topic models* [61, 10, 184, 162, 111, 114, 90], which use a multinomial word distribution to represent a topic, and explain the generation of the text collection with a mixture of such topics. We can see that the topic models advance beyond the mixture language models such as the Poisson mixtures in a way that we can now characterize every particular mixture component as a topic (a latent context).

2.4.1 Probabilistic Latent Semantic Analysis

The earliest probabilistic topic model is known as probabilistic latent semantic analysis (PLSA) [59]. In the context of information retrieval, the model is also called probabilistic latent semantic indexing (PLSI) [60].

The basic assumption of a topic model is that there are k latent topics in the text collection, each of which is represented by a multinomial distribution of words. We use θ_j to denote the multinomial distribution for the j-th topic, over all $w \in V$. We introduce a new parameter $p(\theta_j|d)$ to denote the distribution of selecting a particular topic from the mixture model by a document. $\{p(\theta_j|d)\}_{j=1...k}$ thus makes a multinomial distribution of topics given a particular document. Please note that this distribution is sensitive to individual documents. The log likelihood function of $\mathcal{D}$ can then be rewritten as

$$\log p(\mathcal{D}|\mathcal{M}) \propto \sum_{d\in\mathcal{D}} \sum_{w\in d} \log \sum_{j=1}^{k} p(\theta_j|d)p(w|\theta_j), \tag{2.5}$$

The parameters in Equation 2.4.1 are $\Psi = \{p(\theta_j|d), p(w|\theta_j)\}_{w,j,d}$. By maximizing the log likelihood, we can estimate the parameters of PLSA. Please note that there is no closed form solution for the parameters, the estimation of which is usually done through the Expectation-Maximization algorithm [36]. Please note that the distribution of topics are now estimated from the data instead of pre-specified.

One simply extension to PLSA is to mix the topics with a global background B. This will give us a modified PLSA model as

$$\log p(\mathcal{D}|\mathcal{M}) \propto \sum_{d\in\mathcal{D}}\sum_{w\in d}\log[(1-\lambda_B)\cdot\sum_{j=1}^{k}p(\theta_j|d)p(w|\theta_j)+\lambda_B p(w|B)], \tag{2.6}$$

The advantage of this model is that the common words in English, such as stop words and syntactic words, will be explained by the background context B. Therefore, the words with high $p(w|\theta_j)$ in each topic model will be meaningful content words through which we can interpret the semantics of the topic. This modified PLSA model has been proven to perform well in text mining tasks like [184] and [113].

2.4.2 Latent Dirichlet Allocation

PLSA is widely used in the context of text mining and information retrieval [59, 184, 113]. One criticism of PLSA is that it still has quite a lot of free parameters, so the model is likely to be overfit the data. An approach proposed to solve this is to introduce an additional regularization to the mixture coefficients, so that each multinomial vector $\vec{a}_d$ is sampled from the same Dirichlet distribution.

The new likelihood function of the collection can thus be written as:

$$\log p(\mathcal{D}|\mathcal{M}) \propto \sum_{d\in\mathcal{D}}\int_{\vec{a}_d} p(\vec{a}_d|\alpha)[\sum_{w\in V}\log\sum_{j=1}^{k}a_{dj}\cdot p(w|\theta_j)]d\vec{a}_d. \tag{2.7}$$

Such a model is known as the latent Dirichlet allocation (LDA) in the machine learning literature [10]. Please note that because of the integral, the parameter estimation for LDA cannot be handled by a standard EM algorithm. A more complicated estimation method is needed, such as variational inference [10], Gibbs sampling [51], or expectation propagation [121].

There are many other topic models, either extending PLSA or extending LDA. A good review of probabilistic topic models can be found in [161].

2.5 Topic Models with Context Information

Note that both PLSA and LDA just deal with topics, which are one type of the implicit contexts. The topic models still lack the ability to model the much richer context information other than topics. We still don't know how to integrate situational contexts into a topic model and how the topics themselves are affected by various contexts. Some recent work tries to integrate specific context information into topic models in a case-by-case manner. There are basically two approaches to handle context information in topic models.

One of them is to model context as additional observations in the text, and the other integrates context as dependent variables into the topic model.

One approach of handling context information in text is to treat the context information simply as additional observations besides the words. In this way, one can construct a probabilistic model that not only generates the content (i.e., words) in text, but also the context information.

For example, Li et al. proposed a probabilistic model to detect retrospective news events by explaining the generation of "four Ws[1]" from each news article [91]. Wang et al. proposed a model called "Topic over Time," which generates the time stamps of the time stamps of each document using a mixture of beta distributions [171]. [169] presented a similar model that models the generation of geographic locations. In these approaches, the generation of contexts are conditionally independent with the generation of contents. Therefore, it is difficult to infer the influence of context on content. Different context information are also modeled in a case-by-case manner. A model for one type of context is not extendable to handle other types of context.

An alternative approach is to explicitly integrate context variables into topic models, so that the generative process of topics are now dependent on the context. This is a more natural approach since it explains the influence of context on the generation of content. Among the few examples, the Author-Topic model is proposed in [162], which extends LDA to model the association between authors and topics. [159] extends this work to model the senders and receivers in a collection of emails. Wang et al. [172] added time variable to PLSA so that the model could discover correlated bursty topic patterns. Although this work better captures the influence of different types of context on the content of text, the techniques are tuned for specific contexts and specific tasks and thus cannot be directly applied to deal with other types of context.

2.6 Summary

Statistic language models has been widely used in text mining (e.g., [125, 138, 59, 183, 105, 10, 115]). Early language models assume that all the words in the text collection are generated from the same probabilistic model. Later on, people realized that the generative process of text depends on a lot of hidden factors (i.e., contexts). Mixture language models and probabilistic topic models are proposed to capture these hidden factors in a primitive way.

There is some exploration on incorporating context information into topic models. Some treats context information as additional observations generated by the topic model, and some integrates context variables into the generative process. However, we can see that all these models are trying to integrate specific types

[1] who (persons), when (time), where (locations) and what (keywords)

of context information in an ad hoc way. The proposed models are difficult to be extended to handle other context information. To deal with a new context, one needs to design a new topic model from the scratch.

The discussion in this thesis advances beyond all the previous work, by modeling different types of context in a general way. The framework we propose not only unifies many existing models with context, but also provides a way to design solutions for text mining tasks with new types of context information.

Chapter 3

A Conceptual Framework of Contextual Text Mining

In this chapter, we formally define the problem of contextual text mining, including a formal definition of related concepts, a taxonomy of contexts, and a discussion of basic principles of contextual text modeling, followed by a general conceptual framework of contextual text mining. This general framework serves as a guideline for all the instantiations of contextual text mining.

3.1 A Generative View of Text Mining

In many text mining and machine learning problems, a generative view of text mining is taken. In other words, we make a basic assumption that the observed text data is generated from an underlining probabilistic model, which is either known or uncertain [27, 61, 10]. Such a model could either be as simple as a standard distribution of words, such as a unigram language model [138, 183], or as complicated as a multi-component mixture model which follows a complex dependency structure [27, 10].

In general, such a generative model consists of a set of basic events, which usually corresponds to the observations of language units such as words occurring in the text, and a probabilistic distribution that explains how such observations are sampled from a stochastic process. To the best, such a generative process would be consistent with the process that the text data was originally produced (e.g., the process that an article is written by its author).

A primary task of text mining is then to induct such a generative model based on the observed text data. Usually, we assume that we already know the structure of such a model (e.g., a multinomial distribution, or a hidden Markov model), but do not know the parameters of the model. The induction of the model is then equivalent to estimating the parameters of the model. A common approach is to cast the model induction as an optimization problem, in which the values of the parameters are found to maximize the likelihood that the observed data is generated from the model. This approach is called *maximized likelihood estimation* (MLE). In Chapter 2, we have introduced a variety of generative models of text in literature. In this thesis work, we take the view of generative models as our general methodology for text mining.

3.2 Context Dependent Generative Process

The choice of generative models of text usually corresponds to the understanding that how a textual document is produced, in other words, how the author selects one word versus another. Consider a practice that we want to guess the next word that an author would write. If the author writes with a language that we know nothing but the vocabulary, without any prior knowledge, the best we can do is uniformly select one word from the vocabulary. In other words, we could only assume that all the words in the text document are generated from a uniform distribution. It takes as much as $\log |V|$ bits to guess the next word, where $|V|$ is the size of the vocabulary. If it is a language that we don't even know the vocabulary, it is impossible to guess the next word, or it takes infinite bits to guess the next word [25].

Once we know some prior information about the document, however, we can do a much better job by incorporating stronger assumptions. If we know the author writes the article in English, it is much less difficult to guess the next word, because we know that some English words appear much more frequently than other English words. According to Shannon's claim, it takes as few as 1.3 bits to guess the next character [154]. This is because we know one context of the language - English, so we have much more to say about the selection of words, or formally, the generative process of text. The selection of words may also depend on the author's personal vocabulary/preferences. A physicist and a poet may use very different words to describe the same object, for example, the moon.

Similarly, the content that an author writes is also related to when he is writing this article, what location he is at, what words he has already written, what goal he is writing for, what topic he is writing about, what mood he has, and so on. In other words, our understanding of the model of the text is now much deeper - the content of a document highly depends on all kinds of situations in which the author was written this document. This is to say, we assume that the generation of text data highly depends on the context in which the text was written. An article written in the 18th century may have nothing about computer; the word "soccer" may not appear in a report about a soccer match in Europe. We can see that the notion of context here covers both linguistic context, such as the other words in the same sentence, and situational context, such as time, location, and sentiment.

Context affects the generation of text content. It is thus desirable to represent the corresponding assumption in the generative model of text once we have such context information. A better generative model should take into consideration of the dependency between the generative process of text and the context information. In other words, a good generative model should represent the structure that each text unit is generated based on the context that the unit belongs to. How to incorporate various context information into the generative model of text is the core task of contextual text mining.

3.3 A Formal Definition of Contextual Text Mining

In this section, we give a more formal discussion of context modeling in text data, by formally define the important concepts.

3.3.1 Definitions of Basic Concepts

Let us introduce a few important notations and definitions that will be frequently referred to in the rest of this thesis. We start with the definition of a few core concept of contextual text mining.

We first introduce $\mathcal{D}$ as a notation for the collection of text data. Since the content of text data is commonly formatted with natural language, we first define the language units in text.

Definition 3.1 (Language Unit): We use w to denote a basic ***language unit*** in text. Such a language unit could be a word, a concept, a phrase, a ngram, an entity, or any other units that carries semantic meaning. We refer to the smallest language units that carries semantic meaning as *basic units*. The most common basic units in text are *words* (or *terms* in the context of information retrieval). So we use w to denote a word, or a textual term in the rest of this thesis unless specified. We define a vocabulary, V, as a set of all w in $\mathcal{D}$.

Please note that in some scenarios, there are symbolic tokens that are not in natural language, but also carry semantic meanings, such as the label of a product, the symbol of a gene in biology literature, and the URL of a web page in query logs, etc. In these cases, we relax the definition of language units so that they also cover those symbolic tokens with semantic meanings. We then define a document in the text collection.

Definition 3.2 (Document): A text ***document***, d, in a text collection $\mathcal{D}$ is a sequence of words $w_1 w_2 ... w_{|d|}$, where w_i is a word from a fixed vocabulary V . Following a common simplification in most work in text mining [61, 10], we can represent a document with a bag of words, i.e., $d = \{w_1, w_2, ..., w_{|d|}\}$. We use $c(w, d)$ to denote the occurrences of word w in d.

Definition 3.3 (Context Feature): A ***context feature*** of text, or a *context variable*, X, is an attribute of the text the values of which could define a partition of the text collection $\mathcal{D}$. Such an attribute could either be intrinsic to the content, such as the appearance of a term, or a topic; it could also be an extrinsic feature of text, such as time, location, and authorship. Please note that the value of X could either be discrete, such as $X_{Author} =$ "$JimGray''$, or continuous, such as "$X_{year} < 2006$." We use uppercase X to denote the variable of a context feature, and lowercase x to denote the value of the feature.

Definition 3.4 (Context): A ***context*** of text, c is a meaningful condition of text data that reflects the situation at which the text is produced. Formally, a context c corresponds to a condition defined by the value of a set of context features, i.e., $\{X_1 = x_1, X_2 = x_2, ..., X_N = x_N\}$. We use $\mathcal{C}$ to denote the set of

all meaningful contexts. Following this definition, each context defines a subspace of the text collection $\mathcal{D}$, which consists of all language units that satisfy this condition.

We further define this set of language unites, $\mathcal{D}_c$, as the ***domain*** of a context c. We require that $\mathcal{D}_c \neq \phi$.

We say that c_1 and c_2 ***overlap*** with each other, if $\mathcal{D}_{c_1} \cap \mathcal{D}_{c_2} \neq \phi$. From the definition of context, we can also see that the smallest context covers a single language unit w, and the largest context covers $\mathcal{D}$ itself. A document d also corresponds to a special type of context. In a particular problem of contextual text mining, we only select the meaningful contexts that we are interested in. It is also worth noticing that the various values of a particular context feature can define a set of contexts (e.g., the time context).

Please note that the aforementioned definitions are general enough to capture different approaches to contextual text mining. When a generative view of text mining is taken, we also need a few key definitions related to generative models of text.

Let us first introduce a ***generative model*** $\mathcal{M}$ for $\mathcal{D}$, which is a probability distribution of the observation of language units in $\mathcal{D}$. $\mathcal{M}$ corresponds to the random process that how the language units in $\mathcal{D}$ are generated. When we assume that the basic language units w are generated independently, $\mathcal{M}$ can be characterized with the distribution $\{p(w|\mathcal{M})\}_{w \in V}$. The likelihood of generating $\mathcal{D}$ with the probabilistic model $\mathcal{M}$ is written as $p(\mathcal{D}|\mathcal{M})$. $\mathcal{M}$ is usually used as a representation of $\mathcal{D}$. $\mathcal{M}$ is also called a ***language model***. When $\mathcal{M}$ models context information as well as content, we call it a ***contextual language model***.

Similarly, we introduce the following definitions related to contextual language models:

Definition 3.5 (Context Model): For a context c, a context model $\mathcal{M}_c$ is a probabilistic model which explains the generative process of language units in $\mathcal{D}_c$. Similarly, $\mathcal{M}_c$ is usually used as a representation of the context c, and can be characterized with $\{p(w|\mathcal{M}_c)\}_{w \in V}$.

Definition 3.6 (Topic/Theme): A ***topic*** or a ***theme***, T, is a semantically coherent subject of a discourse in a text collection. In this thesis, we formally define a topic as a latent context in $\mathcal{D}$. We use T to denote the context feature for topics. If a language unit satisfies the condition $T = t$, we say that it belongs to the latent context of topic t. The context model which represents a topic is called a ***topic model***, denoted as θ_t, which is characterized by a probabilistic distribution of words $\{p(w|\theta_t)\}_{w \in V}$. Clearly, we have $\sum_{w \in V} p(w|\theta_t) = 1$. We assume that there are all together k topics in $\mathcal{D}$.

Definition 3.7 (Contextual Pattern): A ***contextual pattern*** in text is defined as a pattern that could be derived based on conditional distributions involving language units and various contexts. In particular, the conditional distributions include the context model $p(w|c)$, the distribution of context given a language unit, $p(c|w)$, and the distribution of one type of context given another type of context, $p(c'|c)$. A ***contextual topic pattern*** is a contextual text pattern in which at least one of the contexts is a topic.

We refer to the interesting conditional distributions as *basic contextual patterns*, and patterns derived from postprocessing the basic contextual patterns as *refined contextual patterns*.

We can see that this definition of contextual pattern covers a wide range of mining products of contextual text mining tasks as instantiations. For example, topic modeling [59, 10] aims at the discovery of conditional distributions of words given topics as a representation of the topics in text; temporal text mining attempts to extract topic life cycles [113], which can be represented by, or can be extracted by refining the conditional distributions of topics given time context and of time given topics; author-topic analysis targets at discovering conditional distributions of topics given a particular author and conditional distributions of words given different authors and topics [162, 114]. We will discuss specific instantiations of contextual patterns in Chapter 4 and in the concrete contextual text mining applications later in this thesis.

When contexts do not overlap, it is clear that all the words in the domain of c are generated based on $\mathcal{M}_c$. However, when w is in the domain of multiple contexts, it may be generated based on any of those contexts. We define c to be the ***active*** context for w, if w is generated using $\mathcal{M}_c$. Clearly, the context distribution $p(c|w)$ defines how likely c is the active context of w. We define all the language units which are actually generated by $\mathcal{M}_c$ as the ***active domain*** of c.

We can also define other useful concepts for contextual text mining. We leave other definitions to specific chapters in the rest of this thesis.

3.3.2 A Taxonomy of Context

The definition of context is quite broad. We can see that the definition unifies many different notions of context, including both the linguistic context and the situational context, as long as it corresponds to an evaluable condition and defines a subspace of the text. Following the definitions, we can give a more in-depth discussion about different ways to categorize contexts.

Explicit Context and Implicit Context

We can divide contexts into explicit context and implicit context. Recall that every context corresponds to a condition with particular values of context variables. Every language unit which satisfies this condition belongs to the domain of this context. In most cases, whether a language unit satisfies the condition is deterministic. For example, whether a document is produced at some time, published at some location, or written by some author are deterministic. The context corresponding to such a condition is called an *explicit context*, such as time, location, and authorship. Some conditions, however, are not deterministic for some language units. For example, whether a word is about a topic, carries a sentiment, or has a high impact

is somewhat implicit. The context corresponding to such a condition is called an *implicit context*. As a result, the domain of an explicit context is well defined; the domain of an implicit context is vague, which we need to infer from the text data. We will introduce the modeling of explicit context in Chapter 5 and the modeling of implicit context in Chapter 6.

Context of Various Granularity

We can also distinguish contexts according to the granularity that a context applies to. Some contexts are larger, the domain of which covers multiple documents, e.g., time and authorship. The largest (but trivial) context is the whole collection. Some contexts are finer, the domain of which only covers a sentence, or even several words, e.g., the words next to the word "mining." Based on the size of the domain, we can categorize a context into a document-level context, a sentence-level context, or a local adjacency context, etc.

Complex Context

Contexts are not always independent. The time contexts follow the structure of a linear chain; every location has its adjacent locations; different people form a social network structure. These dependent contexts make a complex system by themselves. Such a complex structure of contexts introduces important criteria to context modeling. In general, we use *complex context* to denote the structure of contexts, which we will discuss in details in Chapter 7.

3.3.3 Tasks of Contextual Text Mining

Based on the definitions and discussion about contextual text mining, we can introduce the general tasks of contextual text mining. The general tasks of **contextual text mining** include:

1. constructing a reasonable contextual language model $\mathcal{M}$ for the text data;
2. discovering contextual patterns from text; and
3. further analysis based on the context models and contextual patterns.

When topics are involved in the contexts, we can use "contextual topic patterns" to replace "contextual patterns" in task 2) and 3).

Please note that the third task is defined rather broadly, which covers a lot of analysis based on the output of the first three tasks. Once we have the representation of contexts (e.g., the context model), we can summarize a context; we can compute the similarity of different contexts; we can group similar contexts; we can retrieve relevant contexts according to a query. Using different contextual patterns as features, we can

also categorize contexts, score contexts, and rank contexts; we can also compare the meanings of language units, topics, and other patterns across different contexts.

These basic tasks unify many text mining problems with context information involved. Topic modeling [59, 10] tries to construct probabilistic topic models for text, and discover word distributions for topics; temporal text mining tries to extract topic life cycles and evolutionary topic patterns [113]; spatiotemporal topic analysis attempts to model blog articles with temporal and geographic information and to discover the diffusion patterns of topics over time and location [111]; opinion summarization aims at model the mixture of topics and sentiments and compare topics under different sentiment contexts [110]; personalized search aims at modeling the user's search history and predicting the likelihood of clicking a URL by a user when she issues a query [108]. We will introduce a general instantiation of contextual text mining, contextual topic analysis, in Chapter 4.

3.4 Basic Principles of Contextual Text Modeling

We can see that the central task of contextual text mining is to model the context with a contextual language model. All the other tasks rely on the construction of a reasonable contextual language model. However, the construction of the contextual language model is challenging, since there are many possible assumptions of the generative process. Correspondingly, there are many possible models that could incorporate context information. One could assume that text in all contexts are generated in the same way (e.g., from a unitary model); or every word is generated in a different way, according to its previous word; or the generation of text is dependent on time but not on location; etc.

What is a reasonable generative process? What is a reasonable model correspondingly? It is desirable to think about basic principles of incorporating context information in the generative process. Let us think about the simplest case, where there is only one context. In such a case, we can fairly assume that the text is generated from a unitary language model such as a multinomial. Without loss of generality, when multiple contexts are available, we assume that there is a unitary model for each context which explains the generative process of the text in the active domain of this context. If the contexts do not overlap, the domain of every context is also its active domain. The contextual language model is still easy to construct - each word is generated according to the context model whose domain the word belongs to.

It is much more complicated when contexts overlap - the active domain of each context is uncertain. An article on today's newspaper writes about a topic, which may be because the topic is hot at this *time*, or because the topic is closely related to the *local* audience, or because the *author* is authoritative on this

topic. It is now difficult to tell based on which context the article is generated, or to figure out the active domain of each context. To provide general guidance to the construction of contextual language models, we can define a few basic principles that a good contextual language model for text data should satisfy.

Principle 1. Language units in different contexts are generated from different context models.

This is to say, if language unit w is not in the domain of context c, c will not influence the generative process of w. In other words, the generation of w doesn't depend on the context model $\mathcal{M}_c$. If context c_1 and context c_2 does not overlap, the context model $\mathcal{M}_{c_1}$ and $\mathcal{M}_{c_2}$ are likely to be different. For example, scientific topics in the 16th century are different from topics in the 21st century. Please note that w is in the domain of c doesn't mean that w is necessarily generated based on $\mathcal{M}_c$.

Principle 2. A language unit in multiple contexts is generated from any of the context models.

If w is in the domain of a set of contexts, $c_1, c_2, ..., c_k$, where $k > 1$, w could be generated according to the model of any context c_i, where $1 \leq i \leq k$. A word is not necessarily generated from the same context with the other words in the same document. In other words, w could be generated based on any context whose domain it belongs to, and w is generated from which context model is uncertain. For example, a research paper may choose to write about the hottest problem in the current year; it may also choose to write about a relevant topic according to the conference it is submitted to.

Principle 3. Language units in dependent contexts are generated from dependent context models.

This principle is not needed if all contexts are independent with each other. However, in reality there is usually a dependency between contexts. Some contexts overlap with others; some contexts are contained by other contexts; and some contexts are similar to each other (e.g., the year 2000 and the year 2001). We assume that the generative process of text in these dependent contexts are also dependent with each other. Topics in the SIGMOD conferences are similar to topics in the VLDB conferences; queries from Urbana are like queries from Champaign. A desirable contextual language model should capture the dependency between contexts. Based on a reasonable definition of context similarity, this principle can be instantiated as "language units in similar contexts are generated from similar context models."

We assume that a reasonable contextual language model and its corresponding generative process should follow these general principles. In the following section, we will introduce a general framework for contextual

text mining which satisfies these principles.

3.5 The Framework

In this section, we provide the general framework for contextual text mining. The general framework can be divided into three major stages. The basic tasks in the first stage focus on selecting interesting contexts and labeling the domain of each context, which can be regarded as a preprocessing of contextual text mining. The second stage is the phase of context modeling, which consists of the construction of a contextual language model and the extraction of contextual patterns. After that, the third stage makes use of the context models and the contextual patterns extracted to conduct higher-level analysis. This can be considered as a postprocessing step for contextual patterns. A more detailed view of the framework is shown in Figure 3.1.

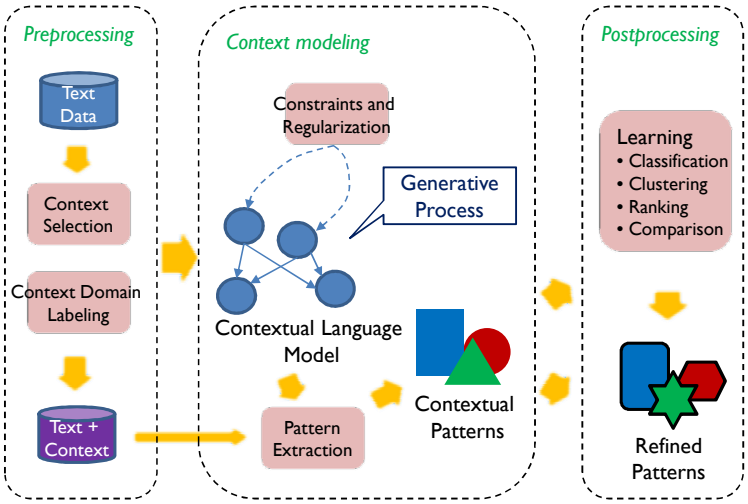


Figure 3.1: A Framework for Contextual Text Mining.

The general process of a contextual text mining task works as follows: in the preprocessing phase, given the text data, the key contexts are identified and the domain of each context is labeled. This will give us a contextualized text collection which is passed to the second phase. In the second phase, the modeling phrase, a contextual language model is constructed based on selection of contexts and the assumptions about the generative process. Constraints and regularization are added onto the model to reflect various assumptions and principles about the generative process and prior knowledge. Once the model is constructed, we extract the contextual patterns (i.e., interesting conditional distributions) either through parameter estimation of the model or through other types of inference (e.g., Bayesian inference). A variety of contextual patterns will be passed to phase three. In the postprocessing phase, the context models and contextual patterns are utilized as a characterization of the data directly, as features for various learning tasks, or as input for the

extraction of refined patterns.

In the rest of this section, we elaborate some important components in this general framework.

3.5.1 The Contextual Language Model

We can see that the core phase in contextual text mining is the modeling step, which constructs the contextual language model and extracts contextual patterns. In general, any probabilistic generative models of text that reflects the context-dependent generative process in Section 3.2 and satisfies the principles in Section 3.4 can be a reasonable contextual language model for contextual text mining. Such a model should include both content variables as well as interesting context variables, based on which we can inference the conditional distributions corresponding to the contextual patterns defined in Section 3.3.1.

What would be a good example of such a contextual language model? In this section, we introduce a probabilistic mixture model as an example of the generative model for contextualized text data. A mixture model is a probabilistic distribution that consists a convex combination of multiple component distributions. In this model, each component is a context model, which is a distribution of language units in the active domain of the context. The mixture model is constructed as a linear combination of all context models.

Formally, the mixture model can be written as

$$p(w|\mathcal{M}) = \int_{\mathcal{C}} h_w(c)p(w|\mathcal{M}_c)dc, \tag{3.1}$$

where w is a particular token of a language unit and $h_w(\cdot)$ is a non-negative function for each w. So $\int_{\mathcal{C}} h_w(c)dc = 1$.

When there are finite number of contexts, Equation 3.1 can be written as a finite mixture model,

$$p(w|\mathcal{M}) = \sum_{c\in\mathcal{C}} a_{c,w}p(w|\mathcal{M}_c), \tag{3.2}$$

where $0 \leq a_{c,w} \leq 1$ and $\forall w, \sum_c a_{c,w} = 1$. $a_{c,w}$, or $h_w(c)$ in general, is a mixture coefficient of the effect of context c on language unit w, which controls the activeness of the context on the language unit. Please note that up to now we haven't made any assumptions on the particular distributions we choose for each context model. When we have multiple language units in the collection, the likelihood that the data collection $\mathcal{D}$ is generated by the model $\mathcal{M}$ will depend on the particular distributions we choose. Each component model could either be a multi-Bernoulli distribution, which models the appearance/absence of each word in the vocabulary; or a multinomial distribution, which models the appearance of each word token in the collection; or a Poisson distribution, which directly models the frequency of each unique word. It could

even represent a more complicated process, such as the Dirichlet compound multinomial distribution [100]. Following the common discussions in text mining [183, 61, 10], in the scope of this thesis, we assume that in each component context model, each word is generated independently from a multinomial distribution. Therefore, $p(w|\mathcal{M}_c)$ is now the probability that the spell of the word token w is generated from the context model $\mathcal{M}_c$, no matter where the word token actually appears.

When we model the generation of every token in the collection, the set of parameters $a_{c,w}$ can be denoted as a matrix $A_{|\mathcal{C}|\times|\mathcal{D}|}$, where $|\mathcal{C}|$ is the number of contexts and $|\mathcal{D}|$ is the size of the data collection. We use vector $\vec{a}_w$ to denote the context coefficients for the language unit w. Similarly, we can also denote the parameters of the context models as a matrix $P_{|\mathcal{C}|\times|V|}$. The likelihood of the observed data can thus be written as

$$p(\mathcal{D}|\mathcal{M}) \propto \prod_{w\in\mathcal{D}} [\sum_{c\in\mathcal{C}} a_{c,w} p(w|\mathcal{M}_c)], \tag{3.3}$$

Here we use w to denote a particular occurrence (or a particular token) of a language unit (e.g., a word) in $\mathcal{D}$. The same word may appear multiple times in $\mathcal{D}$ and have multiple occurrences, and will introduce multiple multipliers in Equation 3.3. Please note that although the mixture coefficient $a_{c,w}$ can be regarded as the expected activeness of context c on word w, it does not necessarily correspond to the actual probability that each token w is generated by $\mathcal{M}_c$, or $p(c|w)$. In fact, it can be associated with any arbitrary value or constraints that reflect the user's general understanding about the generative process, or any other prior knowledge about the effect of contexts. For instance, $a_{c,w}$ can be set uniformly. For a context c that w is not in its domain (i.e., $w \notin \mathcal{D}_c$), we set $a_{c,w} = 0$.

It is clear to see that this mixture model satisfies the first and second principles discussed in Section 3.4. We can incorporate various other constraints or regularization in this model, which allows us to introduce important prior knowledge or assumptions about the generative process and regularize the model so that it satisfies other important principles (e.g., the third principle in Section 3.4). We will introduce various types of constraints later in this section. Once the model is constructed, the rest of the task is to extract various contextual patterns (i.e., the conditional probabilities between language units and contexts) by fitting the contextualized text data.

3.5.2 Regularization and Constraints

In Equation 3.2, both the mixture coefficients and the parameters of each context model are free parameters. The only constraints are the natural sum-to-one constraints as well as the affiliations between words and the

domain of contexts (i.e., $a_{c,w}$ is zero if w is not in $\mathcal{D}_c$, and nonnegative otherwise). When implicit contexts such as topics and sentiments are involved, there isn't even a clear affiliation between words and the domain of contexts.

With so many parameters, the model is likely to overfit the data, which is already quite sparse. In reality we usually include various constraints and regularization to regulate the model, so that the model follows important principles and assumptions about the generative process and the particular contexts. Another importance of adding constraints and regularization is that it helps the model to incorporate various types of prior knowledge of particular contexts and particular applications.

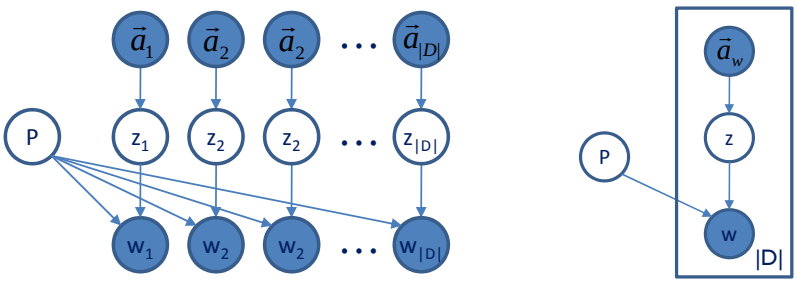
For example, we may assume that the general effect of a document-level context is equivalent on each word in a document. This is to say, a document-level context (e.g., the author) has a general impact on a document, which can be represented by a document-level coefficient $a_{c,d}$. Therefore, we can incorporate this assumption to the model in Equation 3.2, by formally introducing the constraints that

$$\forall d, \forall w_i, w_j \in d, a_{c,w_i} = a_{c,w_j}. \tag{3.4}$$

This is equivalent to replacing the word-level coefficients $a_{c,w}$ with document-level coefficients $a_{c,d}$, and the likelihood function in Equation 3.3 can be rewritten as

$$p(\mathcal{D}|\mathcal{M}) \propto \prod_{d\in\mathcal{D}} \prod_{w\in d} [\sum_{c\in\mathcal{C}} a_{c,d} \cdot p(w|\mathcal{M}_c)]. \tag{3.5}$$

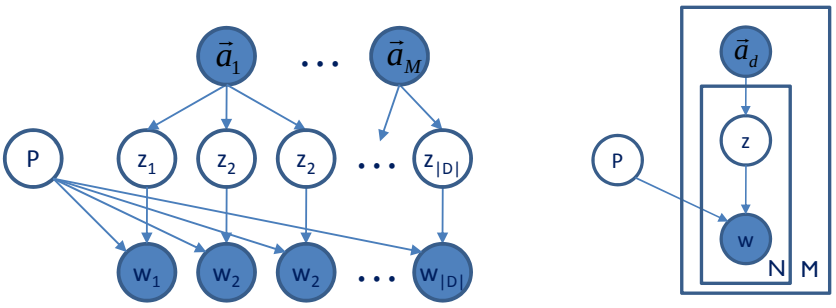
Thinking from another view, adding such a constraint is also equivalent to changing the generative process corresponding to the model. The generative process of the model in Equation 3.2 can be expressed in a graphic model shown in Figure 3.2. With the document-level constraints, the model in Equation 3.2 can be presented in Figure 3.3.



(a) The graphic model. (b) The simplified graphic model.

Figure 3.2: The generative process of the general mixture model (Equation 3.3).

In Figure 3.2 and Figure 3.3, z_i is a latent variable that denotes which context model $\mathcal{M}_c$ the word w_i is



(a) The graphic model. (b) The simplified graphic model.

Figure 3.3: The generative process with document-level constraint (Equation 3.5).

generated from. M is the number of documents in the collection, and $|d|$ is the number of language units in document d. When can see that when the document-level constraints in Equation 3.4 is added to the general mixture model, the corresponding generative process is changed from the one presented in Figure 3.2 to the one presented in Figure 3.3. When all the contexts are latent topics, the generative process in Figure 3.3 is exactly the generative process of the PLSA model. Clearly, we see that a probabilistic topic model like PLSA is the special case of the general contextual mixture model with appropriate constraints are added into the model.

Many other constraints can be added into the model. This will make the model fit for different generative processes, accommodate various prior knowledge, and be regularized according to important principles. Such constraints and regularization can be associated with both the mixture coefficients $A_{|\mathcal{C}|\times|\mathcal{D}|}$ and the context models $P_{|\mathcal{C}|\times|V|}$. For example, we can constrain the context models so that some contexts share the same model; or we can constrain the mixture coefficients so that all vectors $\vec{a}_w$ are sampled from the same distribution; or we can incorporated the constraint that some context have a larger effect than others; or we can constrain on a context model so that it is close to our prior understanding of the context; or we can constraint on the contexts so that similar contexts have similar models. All these constraints will introduce new variations on the top of the general model in Equation 3.2. In the rest of the thesis, we will show particular instantiations of the general model, which can deal with different types of context and solve different real world problems.

3.5.3 Extraction of Contextual Patterns

Once the contextual language model is constructed, the next task is to fit the data with the model and extract interesting contextual patterns. Recall that contextual patterns are either represented by or refined from the conditional distributions with language units and contexts, the core task of contextual text mining is thus to infer these conditional distributions by fitting the text data with the contextual language model.

Taking the contextual mixture model in Equation 3.3 as example, we can see that some of the model parameters correspond to some of the contextual patterns (conditional distributions), for example $p(w|\mathcal{M}_c)$. We can extract such contextual patterns by estimating the corresponding model parameters. Without any constraints or regularization, the set parameters in Equation 3.3 are $\Psi = \{a_{c,w}, p(w|\mathcal{M}_c)\}_{c\in\mathcal{C},w\in V}$. We cast the parameter estimation problem as an optimization problem, in which we want to maximize the likelihood that the data is generated from $\mathcal{M}$. We find the optimal solution of the parameters, $\hat{\Psi}$, by

$$\begin{aligned}\hat{\Psi} &= \arg\max_{\Psi} \log p(\mathcal{D}|\mathcal{M}) \\ &= \arg\max_{\Psi} \sum_{w\in\mathcal{D}} \log \sum_{c\in\mathcal{C}} a_{c,w} p(w|\mathcal{M}_c). \end{aligned} \tag{3.6}$$

Equation 3.6 is referred to as maximum likelihood estimation (MLE) in literature. MLE is not the only way to estimate parameters in the given model. There are also other methods to estimate parameters, for example maximum a posterior (MAP) [110, 183].

Please note that parameter estimation is not the only method to extract the contextual patterns (i.e., the conditional distributions). Some inference methods can bypass the parameter estimation and directly infer the conditional distributions. Examples are sampling-based methods like Markov Chain Monte Carlo (MCMC) and its special case, Gibbs sampling [51].

We can usually solve the optimization problem in Equation 3.6 using the Expectation-Maximization (EM) algorithm. The EM algorithm [36] iteratively computes a local maximum of $\log p(\mathcal{D}|\mathcal{M})$. Specifically, in the E-step, it computes the expectation of the complete likelihood $Q(\Psi; \Psi_n)$, where Ψ_n denotes the value of Ψ estimated in the last (n-th) EM iteration. In the M-step, the algorithm finds a better estimate of parameters, Ψ_{n+1}, by maximizing $Q(\Psi; \Psi_n)$:

$$\Psi_{n+1} = \arg\max_{\Psi} Q(\Psi; \Psi_n) \tag{3.7}$$

Computationally, the E-step boils down to computing the conditional distribution of the hidden variables in the generative process given the data and Ψ_n. The hidden variables usually correspond to which context is active for which language unit. For example, the hidden variables in Equation 3.3 correspond to the events that a term w is actually generated from context c (e.g., c is active for w). Let us denote such an event as "$z_{w,c} = 1$." Formally, we have the **E-Step:**

$$p(z_{w,c} = 1) = \frac{a_{c,w} p(w|\mathcal{M}_c)}{\sum_{c'\in\mathcal{C}} a_{c',w} p(w|\mathcal{M}_{c'})} \tag{3.8}$$

Please note that we can also write the distribution of $p(z_{w,c} = 1)$ as $p(c|w)$. This conditional distribution is also one type of interesting contextual patterns based on the definition in Section 3.3.1. With the distribution of the hidden variables, we can write down the expectation of the complete likelihood of the data:

$$Q(\Psi; \Psi_n) = \sum_{w \in \mathcal{D}} \sum_{j} p(z_{w,c} = 1) \log a_{c,w} p(w|\mathcal{M}_c) \tag{3.9}$$

Given the expectation of the complete likelihood, the maximization problem in the **M-Step** (i.e., Equation 3.7) usually has a closed form solution. For example, the parameters in Equation 3.3 can be computed as follows:

$$a_{c,w} = \frac{p(z_{w,c} = 1)}{\sum_{c'} p(z_{w,c'} = 1)} \tag{3.10}$$

$$p(w|\mathcal{M}_c) = \frac{\sum_{w' \in \mathcal{D}} p(z_{w',c} = 1)\delta(w', w)}{\sum_{w' \in \mathcal{D}} p(z_{w',c} = 1)} \tag{3.11}$$

$\delta(w', w)$ is a function that returns 1 if token w' and w share the same spell, and zero otherwise. Note that for $\forall c, w \notin \mathcal{D}_c$, we initialize $a_{c,w} = 0$. This value won't change after each EM iteration. The EM algorithm will converge at a local maximum of $\log p(\mathcal{D}|\mathcal{M})$. When there is no constraint on $a_{c,w}$, we can easily deduct $a_{c,w} = p(z_{w,c} = 1)$ from Equation 3.10. In this particular case, $a_{c,w}$ is consistent with the actual activeness of context c on w (i.e., $p(c|w)$). In other cases when there are constraints on $a_{c,w}$, the solution of the M step could be quite different.

There is an intuitive interpretation of the parameter estimation. Suppose the contexts do not overlap, or we magically know exactly which words are generated by which context, the estimation of the context models is quite easy - we can simply estimate the model for each context independently from the words in its *active domain*. In an EM algorithm, every E-step is trying to guess the active domain of each context model, and every M-step estimates the context models and mixture coefficients based on the guess. Similar process can be found in other inferencing algorithms such as the Gibbs sampling [51].

Once we find these basic conditional distributions in the contextual language model (e.g., $p(w|c), p(c|w)$), we can infer further other conditional distributions (e.g., $p(c'|c)$).

3.5.4 Postprocessing of Contextual Patterns

There are three basic ways to utilize the contextual patterns extracted from the modeling phrase in real applications. One of the ways is to use the contextual patterns directly to present the characteristics of the text data. For example, one can use the words with largest probability in the context model $p(w|\mathcal{M}_c)$ to

characterize the semantics carried by a context c. This is especially useful when c is an implicit context such as a topic. From the contextual patterns $p(c|w)$ and $p(c'|c)$, we can also tell the effect of a context on a language unit, or on other contexts.

Another usage of the contextual patterns is to postprocess the basic patterns in order to extract refined patterns. For instance, one can compare the conditional distributions $p(w|c)$ for different contexts to summarize the difference between contexts; one can infer the dependency between contexts by calculating the similarity between the context models; one can generate meaningful labels based on the contextual patterns to interpret the semantics of a context (e.g., a topic) to real users. These refined contextual patterns are potentially more interesting and useful in a lot of text mining applications. In Chapter 8 and Chapter 9, we will introduce two applications of postprocessing contextual patterns, to extract refined patterns like evolutionary theme graph and generate meaningful labels for context models.

The third way to utilize contextual patterns is to use them as additional features in machine learning tasks on the text data, such as document retrieval, text classification, and text clustering. In Chapter 10 and Chapter 11, we will introduce two representative examples of utilizing contextual patterns to help text retrieval and web search.

3.6 Summary

In this chapter, we presented a general conceptual framework for contextual text mining. We provide a novel and formal definition of the key concepts and the basic tasks of contextual text mining. We then provide a formal discussion about the taxonomy of contexts, and there basic principles in constructing a desirable contextual language model. In Section 3.5, we presented the conceptual framework of contextual text mining which includes a preprocessing phase, a context modeling phase, and a postprocessing phase.

The core component of contextual text mining is the construction of the contextual language model which explains the generative process of text with dependency on contexts. A desirable contextual language model should reflect the basic principles discussed in Section 3.4. We introduced a good example of such a contextual language model as a general mixture model with the model of every context as mixture components. The model is quite general when no constraint is added onto the model parameters. A component of regularization and constraints is then introduced to guarantee that the mixture model follows the basic principles and incorporates prior knowledge. With specific constraints on the model parameter, we have shown that a probabilistic topic model such as PLSA is a special case of the contextual mixture model. We then presented how to extract interesting contextual patterns by fitting the data with the contextual language model, and

how to postprocess the contextual patterns to facilitate text mining tasks.

The conceptual framework is very general. There is still a gap between the conceptual framework and real world applications. In next chapter, we will introduce a functional instantiation of the general framework, which generally models the contextual text mining problems with topics involved, the instantiations of which are already proved effectively in a variety of real world applications.

Chapter 4

Contextual Topic Analysis: A Functional Framework

In Chapter 3, we introduced a very general conceptual framework for contextual text mining. This general framework provides the guideline for particular contextual text mining tasks, by preprocessing contextualized text data, constructing contextual language models, extracting contextual patterns, and postprocessing contextual patterns. Both the framework and the definition of the contextual language model is quite general. A lot of contextual text mining tasks can be shown as the instantiation of this framework and be accomplished by some instantiation of the contextual language model.

Since the framework is so general, we need a bridge between the framework and real world applications. The core phase of the framework is the context modeling, in which the contextual language model is constructed and the basic contextual patterns are extracted. We also need an instantiation of the contextual language model which is functional to carry on mining tasks directly, but also general enough to handle different types of context. To make this conceptual framework functional and easily applicable to real text mining applications, we introduce a functional instantiation of this framework, contextual topic analysis, which refers to contextual text mining with topics and additional types of context. We introduce a functional instantiation of the contextual language model by extending a probabilistic topic model (i.e., PLSA) with context variables (see [114]). The contextual topic model we propose, called contextual probabilistic latent semantic analysis (CPLSA), is both practical and general enough. It covers a number of powerful special versions, with which we can fulfill a lot of contextual text mining tasks with topics involved.

4.1 Overview

Many text mining tasks involve the extraction of topics from text [61, 10, 184, 113, 171, 111, 115]. A topic is a latent context the domain of which is uncertain. Words generated based on the same topic are likely to capture similar semantic meanings.

If topics are the only meaningful context in a text collection, the modeling of topics can be handled by probabilistic topic models, such as PLSA [61] and LDA [10]. We have already shown in Section 3.5.1 that

a topic model like PLSA is a special instantiations of the general contextual mixture model (Equation 3.2). When other types of context information are involved in the text mining task, the problem becomes much more challenging. In this chapter, we show how to model topics and other types of context in the contextual language model, by constructing a contextual topic model.

A text document is often associated with various kinds of context information, such as the time and location at which the document was produced, the author(s) who wrote the document, and its publisher. The topics of text documents with the same or similar context are often correlated in some way. For example, news articles written in the period of some major event all tend to be influenced by the event in some way, and papers written by the same researcher tend to share similar topics.

In order to reveal interesting topic patterns in such contextualized text data, it is necessary to consider context information when analyzing the topics covered in such data. Indeed, there have been several recent studies in this direction. For example, the time stamps of text documents have been considered in some recent work on temporal text mining [72, 137, 113, 20]. Also, author-topic analysis is studied in [162], and cross-collection comparative text mining is studied in [184]. All these studies consider some kinds of context information, i.e., time, authorship, and subcollection.

Time, authorship, and subcollection are by no means the only possible context information of a document besides topics. In fact, any metadata entry of a document can indicate a context and all documents with the same value of this metadata entry can be considered as in the same context. For example, the source of a news article, the author's age group, occupation, and location of a weblog article, and the citation frequency of a research paper, are all reasonable context information. Moreover, a document may belong to multiple contexts. By analyzing the variations of topics over these contexts, a lot of interesting text mining tasks can be addressed, such as spatiotemporal text mining, author-topic evolutionary analysis over time, and opinion comparison over different age groups and occupations.

However, existing techniques are usually tuned for some specific tasks, and are not applicable to consider other kinds of contexts. For example, one cannot directly use the temporal text mining techniques to model the occupation of authors. This indicates a serious limitation of existing contextual analysis of topics: every time when a new combination of context information is to be considered, people have to seek for solutions in an ad hoc way.

Therefore, it is highly desirable to introduce a general solution for contextual topic analysis, which is the instantiation of contextual text mining with topics involved in the interesting contexts. It is desirable to derive a model that is highly general to conduct the common tasks of these specific contextual topic analysis problems, and easy to be applied to each of them with appropriate regularization.

In this chapter, we study this instantiation of the general contextual text mining, by defining **Contextual Topic Analysis** and its common tasks. We extend the probabilistic latent semantic analysis (PLSA) model to incorporate context information, and develop a contextual probabilistic latent semantic analysis (CPLSA) model to facilitate contextual topic analysis in a general way. By fitting the model to the text data to mine, we can (1) discover the global salient topics from the collection of documents; (2) analyze the content variation of the topics in any given view of context; and (3) analyze the coverage of topics associated with any given context.

These tasks are general and can be easily applied to different specific contextual topic analysis problems. In this chapter, we show that many existing contextual topic analysis problems can be defined as special cases of contextual topic analysis, and can be solved with regularized versions of the mixture model we proposed, corresponding to the context information and the mining tasks it involves. Although it may not be the only possible model for contextual topic analysis, the model is quite flexible to adapt different assumptions.

4.2 Contextual Topic Analysis

Given a collection of documents, we assume that there is a set of context in additional to a set of topics covered in the collection. Our goal is generally to conduct context-sensitive analysis of these topics.

Formally, we assume that there are altogether k major topics in our collection, $\Theta = \{\theta_1,..., \theta_k\}$. In addition to the set of topics, there are a set of n other contexts, possibly overlapping, either explicit of implicit, in the collection $\mathcal{D}$. We notate these contexts with $\mathcal{C} = \{c_1, ..., c_n\}$. Suppose $\mathcal{D} = \{d_1, ..., d_{|\mathcal{D}|}\}$ as a collection of documents. We introduce $\mathcal{C}_i \subseteq \mathcal{C}$ as a subset of contexts the domain of which contains document d_i. A document belongs to multiple such contexts.

In contextual topic analysis, our goal is to analyze the context-sensitive patterns of topics in such a text collection. Formally, we are interested in the conditional distributions with topics involved. Apparently, these contextual topic patterns include conditional distributions such as $p(w|\theta_i)$, $p(\theta|c)$, and $p(w|\theta_i, c)$. In other words, we would like to discover the semantics of the k major topics (i.e., $p(w|\theta_i)$) and how they vary according to other types of contexts both in terms of content (i.e., $p(w|\theta_i, c)$) and strength (i.e., $p(\theta|c)$).

For example, if the context variable we are interested in is time, we will assume that there is a potentially distinct "version" of the k topics in every different time period; such different "versions" of the same topics model the semantic variations of those topics over time. We formally define such a version of topics as a *View* of topics.

Definition 4.1 (View) A ***view*** of topics in a contextualized text collection $\mathcal{D}$, notated as v_i, is a set of

topic models $\theta_{i1}, ..., \theta_{ik}$, where every θ_{il} is a multinomial distribution of words corresponding to the topic θ_l according to the view v_i.

We assume that there are n views in our collection, $v_1, ..., v_n$, where a view v_i corresponds to a context c_i. Therefore, a document is assumed to potentially have multiple views; precisely which views are taken depends on the document and its context. Each view v_i can be taken by any document in the context c_i.

This is to say, although there are only k major topics in the text collection, every topic has n topic models, each of which corresponds to the variant semantics of that topic under a context c_i. In other words, the overlap of a topic θ_l and a context c_i makes a refined context (corresponding to this topic under this context), the context model of which is θ_{il}.

For example, the meaning of the general topic "information retrieval" is changing at different time period. Decades ago, this topic focus on the index of documents and heuristic term weights, while nowadays the same topic focuses more on learning models and web scale retrieval systems.

Not only the meaning of topics varies in different contexts, but also their strength. There are much more papers on the topic "web mining" today than 10 years ago. "Hurricane" is a hot topic in coastal states rather than in midwest. We further introduce the concept of *coverage* to model the strength of topics in various contexts.

Definition 4.2 (Coverage) A ***coverage*** of topics, denoted as κ_j, is a distribution over the k major topics $p(l|\kappa_i)$ $(l \in \{1, ..., k\})$. Clearly, $\sum_{l=1}^{k} p(l|\kappa_j) = 1$.

Like views, We als assume that there are n topic coverages in our collection, κ_1, ..., κ_n, each κ_i which corresponds to a context c_i. Some contexts may share the same coverage. In that case, the $\kappa's$ corresponding to those context will be identical. In such a case, we also assume that there are m *distinct* coverages in the collection, and use $c(\kappa_j)$ to denote the contexts which share the same coverage κ_j.

The latent structure of topics, views and coverages in a contextualized document collection is illustrated in Figure 4.1.

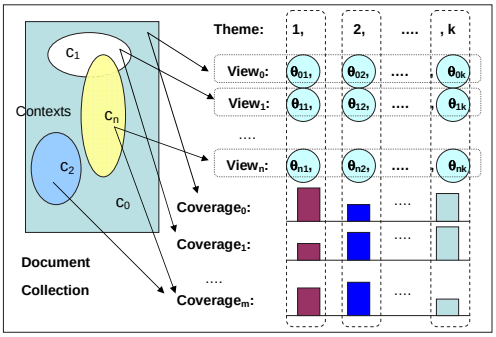


Figure 4.1: The topic-view-coverage structure in a text collection

Topic models, views, and coverages thus consist the basic contextual topic patterns in contextual topic analysis. The major task of **contextual topic analysis** can be defined as to recover the n views, $v_i = (\theta_{i1}, ..., \theta_{ik})$, $i = 1, ..., n$, and n topic coverages $\kappa_1, ..., \kappa_n$ (including m distinct coverages) from the collection $\mathcal{D}$. There are many different ways to analyze the extracted views and topic coverages. Below we discuss a few interesting cases.

1. Topic extraction: We may extract the global salient topics. Although each topic θ_l varies in different contexts, it is also beneficial to have an explicit model for θ_l in a global view. Basically, this will give us the common information that is shared by all the variations of θ_l in all different contexts. In practice, we can always include a global view v_0, which corresponds to a global context c_0 so that $\mathcal{D}_{c_0} = \mathcal{D}$.

2. View comparison: We may compare the n views. The comparison of a topic θ_l from different views usually represents the content variation of θ_l corresponding to different contexts. By comparing θ_{il} for each view v_i which corresponds to context c_i, we can analyze the influence of the context c_i on the contents of θ_l.

3. Coverage comparison: We may compare the m distinctive coverages. The variations of $p(l|\kappa_j)$ can tell us how likely θ_l is covered by the context(s) corresponding to κ_j. We can analyze how closely a topic is associated to a context, or how context-sensitive a topic is.

Among these cases, 2 and 3 are the most important, which distinguish contextual topic analysis from the traditional topic modeling task, and the application of them facilitate other types of analysis.

With the definition of the general problem of contextual topic analysis, we can show that some specific contextual text mining problems are special cases of contextual topic analysis. For example, in temporal text mining, each context is a time period. The goal of temporal text mining is mainly to compare the coverage variation over different time periods (e.g., theme life cycles in [113]), and sometimes also the content variation of topics over different views, (e.g., evolutionary theme pattern in [113]). In author-topic analysis, each context is an author. We are interested in comparing the content variation of topics over different views (authors) [162].

4.3 Contextual Topic Models

In this section, we propose an instantiation of the contextual language model in Section 3.5.1 in contextual topic analysis. This model, which is a *contextual topic model*, is a contextualized extension of the Probabilistic Latent Semantic Analysis (PLSA) model [59, 61]. Therefore it is referred to as the Contextual Probabilistic Latent Semantic Analysis (CPLSA) model. Our main idea is to allow a document to be generated using multiple views and multiple coverages. The views and coverages actually used in a document usually depend

on the contexts which cover the document. The contexts could be the time or location where the document is written, the source from which the document comes, or any other document-level conditions. We first propose the general CPLSA model, and then introduce two simplified versions of this model that are especially suitable for two representative tasks of contextual topic analysis.

4.3.1 The CPLSA Model

In CPLSA, we assume that document d is generated by generating each word in it as follows: (1) Choose a view v_i according to a view distribution $p(v_i|d)$. (2) Choose a coverage κ_j according to a coverage distribution $p(\kappa_j|d)$. (3) Choose a topic l according to the coverage κ_j. (4) Generate a word using the model of the topic l in the view v_i, θ_{il}.

Formally, the log-likelihood of the whole collection is

$$\log p(\mathcal{D}) = \sum_{d\in\mathcal{D}}\sum_{w\in V} c(w,d)\log(\sum_{i=1}^{n} p(v_i|d)\sum_{j=1}^{m} p(\kappa_j|d)\sum_{l=1}^{k} p(l|\kappa_j)p(w|\theta_{il})) \tag{4.1}$$

The parameters in CPLSA include the view selection probabilities for each document ($p(v_i|d)$), the coverage selection probabilities for each document ($p(\kappa_j|d)$), the coverage distributions ($p(l|\kappa_j)$), and the topic models in each view ($p(w|\theta_{il})$).

It is easy to show that CPLSA is an instantiation of the general contextual mixture model in Equation 3.2 with specific constraints. In CPLSA, we actually have a total of $n \times k$ refined contexts (the overlap of every topic and every context makes the domain of a refined context), thus have $n \times k$ component multinomial distributions of words. In other words, comparing with the general contextual mixture model in Equation 3.2, we have $n \times k$ context models. However, each unitary context in CPLSA is in fact the overlap of a topic and another context (θ_l and c_i). Since all the contexts other than topics are at document-level, let us compare the CPLSA model with the general model in Equation 3.5. This is to say, every θ_{il} corresponds to a $\mathcal{M}_c$ in Equation 3.5, and every $\alpha_{d,c}$ in Equation 3.5 corresponds to $p(v_i|d)\sum_{j=1}^{m} p(\kappa_j|d)p(l|\kappa_j)$ in Equation 4.1.

While we can potentially use all the views to generate a document, often the generation of a particular document d only involves a subset of these views because it only belongs to a few contexts. More specifically, the view selection distribution $p(v_i|d)$ determines which views will actually be used when generating words in document d. This distribution would assign zero probabilities to those views that are not selected. For example, if the views that we are to model correspond to the temporal context of a document and we have one global view spanning in the entire time period, then a document at time point t_i would be generated using two different views – the view corresponding to time point t_i and the global view, which is applied to

all the documents.

Orthogonal to the choice of views, we also assume that we have choices of topic coverage distributions. The different coverage distributions are to reflect the uneven coverage of topics in different contexts and to capture the common coverage patterns. For example, if we suspect that the coverage may vary depending on the location of the authors, we can associate a particular coverage distribution to each location, which will be shared by all the documents in the location. After we learn such coverage distributions, we can then compare them across different locations. Once again, exactly which coverage distributions to use would depends on which contexts the document belongs to.

The mixture model can be fit to a contextualized collection $\mathcal{D}$ using a maximum likelihood estimator. The EM algorithm [36] can be used in a straightforward way to estimate the parameters; the updating formulas are as follows:

$$p(z_{w,i,j,l}=1) = \frac{p^{(t)}(v_i|d)p^{(t)}(\kappa_j|d)p^{(t)}(l|\kappa_j)p^{(t)}(w|\theta_{il})}{\sum_{i'=1}^{n} p^{(t)}(v_{i'}|d)\sum_{j'=1}^{m} p^{(t)}(\kappa_{j'}|d)\sum_{l'=1}^{k} p^{(t)}(l'|\kappa_{j'})p^{(t)}(w|\theta_{i'l'})}$$

$$p^{(t+1)}(v_i|d) = \frac{\sum_{w\in V} c(w,d)\sum_{j=1}^{m}\sum_{l=1}^{k} p(z_{w,i,j,l}=1)}{\sum_{i'=1}^{n}\sum_{w\in V} c(w,d)\sum_{j=1}^{m}\sum_{l=1}^{k} p(z_{w,i',j,l}=1)}$$

$$p^{(t+1)}(\kappa_j|d) = \frac{\sum_{w\in V} c(w,d)\sum_{i=1}^{n}\sum_{l=1}^{k} p(z_{w,i,j,l}=1)}{\sum_{j'=1}^{m}\sum_{w\in V} c(w,d)\sum_{i=1}^{n}\sum_{l=1}^{k} p(z_{w,i,j',l}=1)}$$

$$p^{(t+1)}(l|\kappa_j) = \frac{\sum_{d\in\mathcal{D}}\sum_{w\in V} c(w,d)\sum_{i=1}^{n} p(z_{w,i,j,l}=1)}{\sum_{l'=1}^{k}\sum_{d\in\mathcal{D}}\sum_{w\in V} c(w,d)\sum_{i=1}^{n} p(z_{w,i,j,l'}=1)}$$

$$p^{(t+1)}(w|\theta_{il}) = \frac{\sum_{d\in\mathcal{D}} c(w,d)\sum_{j=1}^{m} p(z_{w,i,j,l}=1)}{\sum_{w'\in V}\sum_{d\in\mathcal{D}} c(w',d)\sum_{j=1}^{m} p(z_{w',i,j,l}=1)}$$

However, since the model has many parameters and has a high-degree of freedom, fitting it with a maximum likelihood estimator, in general, would face a serious problem of multiple local maxima. Fortunately, in contextual topic analysis, we almost always associate them with appropriate partitions of context. As a result, the model is often highly constrained. For example, if all we are interested in is to compare non-overlapping views across different time, then $p(\kappa_j|d)$ becomes a delta function, i.e., $p(\kappa_j|d) = 1$ if and only if κ_j is the coverage distribution for the time context of d, and $p(\kappa_j|d) = 0$ for all other κ_j's.

Even with such constraints, the model may still have many free parameters to estimate. One possibility is to add some parametric constraint such as assuming all coverage distributions are from the same Dirichlet distribution as done in LDA [10], which would clearly reduce the number of free parameters; indeed, we can easily generalize our model in the same way as LDA generalizes PLSA [61]. However, one concern with such a strategy is that the parametric constraint is artificial and may restrict the capacity of the model to extract discriminative topics, which is our goal in contextual topic analysis. Another approach is to further regularize the estimation of the model by heuristically searching for a good initial point in EM; specific heuristics would depend on the particular contextual topic analysis task.

4.3.2 Special Cases of CPLSA

Although the contextual topic model itself is an instantiation of the contextual language model in Section 3.2, it is still general enough to capture many different tasks of contextual text mining with topics involved. In other words, the CPLSA model is a bridge between the conceptual framework and the real applications of contextual text mining.

Indeed, if we think about specific constraints to the model parameters in CPLSA, we can see that the contextual topic model can instantiate many interesting special cases.

First, if we assume that there is only one context besides topics - the global context, there is only one view and one coverage. So both the view selection distribution and the coverage selection distribution are trivial. The CPLSA model is instantiated as

$$\log p(\mathcal{D}) = \sum_{d\in\mathcal{D}}\sum_{w\in V} c(w,d)\log(\sum_{l=1}^{k} p(\theta_l)p(w|\theta_l)) \tag{4.2}$$

This model is know as a simple mixture model of topics. However, the document boundaries are not utilized in this model. The text collection is treated as a flat bag of words.

Another interesting special case of CPLSA considers every document itself as a context in the text collection and there are no other contexts other than topics and documents. Let us further assume that there is only one unique view of topics. In other words, the model for a topic in every context (document) is the same. So we can simply use θ_l to denote θ_{il}; and the view selection distribution is trivial. Apparently, the document contexts do not overlap. Therefore, there are still M topic coverages in the collection, each of which corresponds to a unique document (assume M is the number of documents). Words in a document can only choose one coverage, the coverage corresponding the the document context itself. Therefore, the CPLSA model is instantiated as

$$\log p(\mathcal{D}) = \sum_{d\in\mathcal{D}}\sum_{w\in V} c(w,d)\log(\sum_{l=1}^{k} p(l|\kappa_d)p(w|\theta_l)), \tag{4.3}$$

which is in fact the original PLSA model.

Considering that the most important tasks of contextual topic analysis are view comparison and topic coverage comparison across contexts, as discussed in Section 4.2, we can introduce another two special cases of CPLSA, which are particularly useful to do these two tasks.

We first introduce the special version of CPLSA to facilitate view comparison. In some cases, we are only interested to model the content variation of topics across contexts, e.g., when we are analyzing the topic evolutions over time [113], or comparing the common topics and corresponding specific topics across

subcollections [184]. In these cases, we can fairly assume that the topic coverage over contexts is fixed, thus does not depend on the contexts that a document is in. Under this assumption, we assume that there is only one coverage κ applicable to each document, the log-likelihood function can be written as

$$\log p(\mathcal{D}) = \sum_{d \in \mathcal{D}} \sum_{w \in V} c(w,d) \log(\sum_{i=1}^{n} p(v_i|d) \sum_{l=1}^{k} p(l|\kappa_d) p(w|\theta_{il})) \tag{4.4}$$

where κ_d is the coverage associated with the document d. We call this simplified version of model as **fixed-coverage contextual mixture model (FC-CPLSA)**. If we have three views, where one is the global view and the other two correspond to subcollections, it will allow us to compare the common topics and specific topics in the two views, as discussed in [184]. If each view corresponds to a time stamp, this model will allow us to analyze the content evolutions of topics over time, as discussed in [113].

In some other cases, we are only interested to model the variation of topic coverage over contexts, e.g., when we are analyzing the life cycles (i.e., strength variations over time) of topics. In these cases, we are not interested in the content variation of local topics, and thus make the assumption that different views of topics are stable. With this assumption, we can simplify the model likelihood as

$$\log p(\mathcal{D}) = \sum_{d \in \mathcal{D}} \sum_{w \in V} c(w,d) \log(\sum_{j=1}^{m} p(\kappa_j|d) \sum_{l=1}^{k} p(l|\kappa_j) p(w|\theta_l)) \tag{4.5}$$

where $p(w|\theta_l)$ is the global word distribution of topic l, which does not vary across contexts. We call this simplified model as **fixed-view contextual mixture model (FV-CPLSA)**. If the only context feature is time, we have two types of coverage distributions κ_d and κ_t, where κ_d is the coverage distribution corresponding to each document and κ_t is the topic coverage for each time period. This will allow us to model the topic life cycles, as introduced in [113]. If we have two context features, time and location, and each context is a combination of time stamp and location, we also have two groups of topic coverage distributions, κ_d and κ_{tl}. This will allow us to analyze the spatiotemporal topic distributions in a spatiotemporal text mining framework.

Please note that in CPLSA, we allow the selection of views and topic coverages as two independent selection process. In other words, the selection of a view according to a context doesn't necessarily mean that one must select the coverage according to the same context. This is reasonable. Consider the process of writing a scientific paper, the author may select a coverage according to the focus of a conference, so that he will cover more about "information retrieval" and less about "computational biology." When comes to the real content he writes about "information retrieval," he will consider the time context and mention more

words that are bursting recently in the topic "information retrieval."

We can however impose the constraint and tie the selection of views and coverages. In this case, there are still multiple views and multiple coverages, but there is only one selection process to select a context. Once the context is selected, the view and the coverage corresponding to this context is automatically selected. The new log likelihood function can be then written as

$$\log p(\mathcal{D}) = \sum_{d\in\mathcal{D}}\sum_{w\in V} c(w,d)\log(\sum_{i=1}^{|\mathcal{C}|} p(c_i|d)\sum_{l=1}^{k} p(l|c_i)p(w|\theta_{il})). \tag{4.6}$$

We refer to this special version of CPLSA as **Tied-CPLSA**.

With these special versions, the CPLSA model can be easily applied to solve a broad family of text mining problems with contextual topic analysis. We will introduce particular applications of the contextual topic model on explicit contexts, implicit contexts, and complex contexts in latter chapters of this thesis.

4.4 Adding Priors to the Contextual Topic Model

In the previous section, we presented how to add in specific constraints to the CPLSA model, so that we can get a variety of special versions of the general model. These constraints correspond to our understanding and assumptions about the contexts and the generative process of the data. The presented constraints are mostly hard constraints, where some model parameters are forced to take specific values (prior knowledge about a context), or some model parameters are constricted to be identical (dependency between contexts).

In many scenarios, we would like to constrain the model in a soft way. We have some prior knowledge about the contexts and the models, but the prior knowledge is not complete, or not certain. On one hand, we want to incorporate the prior knowledge in to the contextual language model, but on the other hand we want the model to have the flexibility to listen to the data.

This is especially useful with implicit contexts, e.g., topics and sentiments, the domain of which is not deterministic. For example, one may know a topic about "iphone" is likely to mention the words "iphone," "apple," "at&t," and "3gs," she may not know the exact word distribution of this topic, or the probabilities $p(w|\theta)$ for other words. In this scenario, we couldn't add hard constraints to the model parameters (e.g., $p(\text{“}apple''|\theta) = 0.05$). Instead, we should allow the model to adjust our prior belief by listening to the data.

Rather than directly constrict the context models with particular values, we define a prior distribution on the context models with the prior knowledge we have about the context. The parameters of the models are then estimated using the maximum a posterior (MAP) estimator instead of the maximum likelihood estimator (MLE). This way would allow us to adopt our prior knowledge about contexts in the contextual

language model, but also allow the model to learn from the data and further improve the accuracy of the context models.

Formally, let Λ denote the parameter of the contextual language model (e.g., CPLSA), we can estimate the model parameters using

$$\hat{\Lambda} = \arg\max_{\Lambda} p(\mathcal{D}|\Lambda)p(\Lambda). \tag{4.7}$$

The EM algorithm will be changed correspondingly. Another interesting problem is how to construct the prior distribution $p(\Lambda)$ based on our prior knowledge about the contexts. We defer this detailed discussion to Chapter 6, in which we introduce a particular instantiation of the contextual topic model to model implicit contexts.

4.5 Regularizing the Contextual Topic Model

Similarly, it is desirable to relax the hard constraints on the dependency between contexts. In Section 4.3.2, if we assume that some contexts are closely related, we set the context models for these contexts identical. For example, we may assume that the word distribution of a topic doesn't change over adjacent two years. However, it is more reasonable to relax this constraint into "the word distribution of a topic doesn't change significantly over adjacent two years." This is especially important to handle complex contexts like social networks and incorporate the principle of similar contexts (i.e., the third principle discussed in Section 3.4) into the contextual language models.

We propose a novel method to integrate the assumptions about the dependency relation between contexts as an explicit regularization term for the likelihood function. In this way, we cast the parameter estimation of the contextual language model to an optimization problem, where we change the maximum likelihood estimator (or the maximum a posterior) into the optimization of an objective function with both the likelihood function and the regularization term.

Formally, we define a regularized data likelihood as

$$O(\mathcal{D}, G) = (1 - \lambda)L(\mathcal{D}) + \lambda R(\mathcal{D}, \mathcal{C}) \tag{4.8}$$

where $L(\mathcal{D})$ is the likelihood (or log-likelihood) of the collection $\mathcal{D}$ to be generated by the contextual language model, and $R(\mathcal{D}, \mathcal{C})$ is a regularizer involving the model parameters and defined based on the dependency structure of the contexts $\mathcal{C}$. The key issue here is how to construct a reasonable regularizer based on the

assumptions about the dependency between various contexts. We defer the detailed discussion to Chapter 7, in which we introduce a particular instantiation of the contextual topic model to model complex contexts.

4.6 Summary

In this chapter, we introduced an important instantiation of the conceptual framework in Chapter 3, by presenting a study of the general problem of contextual topic analysis. It is an instantiation of contextual text mining with topics involved in the contexts. We formally defined the basic tasks of contextual topic analysis, and proposed a novel contextual topic model to extract topics and model their content and coverage variations over different explicit contexts. Although the contextual topic model itself is an instantiation of the general contextual language model, the proposed model is still quite general and cover a family of specific probabilistic models as special cases. The contextual topic analysis and the contextual topic models presented in this chapter serves as a functional framework of contextual text mining, which bridges the general conceptual framework and real applications.

The proposed model, CPLSA, is a special case of the general contextual mixture model, with both topics and other types of contexts. By further regularizing the model parameters, we introduce several special cases of CPLSA, including the Fixed-View CPLSA and the Fixed-Coverage CPLSA. These two instantiations are proved to be effective in real world applications of contextual topic analysis. In the following chapters, we will use real applications of contextual topic analysis to prove the effectiveness of instantiations of the CPLSA model. We will show the power and flexibility of the contextual topic model in handling various types of contexts, including explicit contexts, implicit contexts, and complex contexts.

Chapter 5

Contextual Topic Analysis with Explicit Context

In the previous chapters, we presented a highly general conceptual framework for contextual text mining (Chapter 3), and a functional framework of contextual topic analysis (Chapter 4) with a general contextual topic model called CPLSA. On one hand, CPLSA is still quite general, which unifies a family of models as special cases with different contexts involved and with various constraints added to the model parameters. On the other hand, CPLSA is fully functional, the instantiations of which have already achieved good performance in a wide range of applications.

In this chapter, we present three different real applications of contextual topic analysis. All the three tasks involve the context of topics, as well as one or more additional types of explicit contexts. Specifically, the spatiotemporal topic analysis [111] involves time and geographic location, the temporal-author-topic analysis [114] involves time and authorship, and the event-impact analysis [114] involves the occurrence of events as the context. We apply either the general CPLSA model or special cases of the CPLSA model to the three text mining tasks and evaluate the performance using different real-world datasets. Empirical results show that the CPLSA model and its special versions can effectively model the topics and extract a variety of contextual topic patterns.

5.1 Spatiotemporal Topic Analysis

5.1.1 Overview

With the quick growth during recent years, *weblogs* (or *blogs* for short) have become a prevailing type of media on the Internet [49]. Simultaneously, increasingly more research work is conducted on weblogs, which considers blogs not only as a new information source, but also as an appropriate testbed for many novel research problems and algorithms [76, 167, 53, 52]. We consider weblogs as online diaries published and maintained by individual users, ordered chronologically with time stamps, and usually associated with a profile of their authors. Compared with traditional media such as online news sources (e.g., CNN online) and public websites maintained by companies or organizations (e.g., Yahoo!), weblogs have several unique

characteristics: 1) The content of weblogs is highly personal and rapidly evolving. 2) Weblogs are usually associated with the personal information of their authors, such as age, geographic location and personal interests[77]. 3) The interlinking structure of weblogs usually forms localized micro communities, reflecting relations such as friendship and location proximity [77].

With these characteristics, weblogs are believed to be appealing for research across multiple domains to answer questions such as "what happens over time", "how communities are structured and evolving", and "how information diffuses over the structure". Specifically, there are currently two major lines of research on blog analysis. One line is to understand the interlinking structures (i.e. communities) and model the evolution of these structures. Kumar and others [77] introduced the distribution of blogs over locations and studied how they form communities. They also proposed a way to discover bursty evolution of these communities [76]. The other line is to perform temporal analysis on blog contents and model information diffusions among blogs. Gruhl and others [53] proposed a model for information propagation and categorized diffusing topics into chatter and spikes; they followed up to prove these temporal patterns of topics are useful to predict spike patterns in sales ranks [52].

Although these existing studies have addressed some special characteristics of weblogs, such as community structures, rapid evolution and time stamps, none of them has addressed well the following two needs in analyzing weblogs.

1. Modeling mixture of subtopics: The content of weblogs often includes personal experiences, thoughts and concerns. As a result, a blog document often contains a mixture of distinct subtopics or themes. For example, a blog article about the topic "Hurricane Katrina" may contain different aspects of concerns (i.e., different topics) such as "oil price" and "response of the government" or even some other topics. In many applications, extracting and analyzing such themes of an event are highly desirable. For example, a news analyzer may ask "what are the growing concerns of common people about Hurricane Katrina", while a marketing investigator may be interested in "what features of iPod do people like or dislike most". To answer such questions, we must analyze the *internal* topic components within a blog article. Unfortunately, most existing work (e.g., [53, 50]) assigns a blog article to only one topic, which has not attempted to model such a mixture of subtopics within a blog document and is thus unable to perform accurate fine-granularity subtopic analysis.

2. Spatiotemporal content analysis: In addition to the available time stamps, a considerable proportion of weblogs are also associated with the profiles of their authors which provide information about their geographic locations. In general, the content information of weblogs may be related with or depend on both the time stamp of the article and the location of its author. For example, when analyzing the spikes

of topics, it is very likely to have a considerable gap of time between the spikes of discussion on the new book "Harry Potter" in England and in China. Therefore, using the average temporal pattern of the book discussion to predict the sales in a specific location would not be reasonable. The public opinions may also be location-dependent. For example, people outside the United States may be concerned more about "shipping price" and "repair warranty" of IBM laptops than those inside the United States, and the public responses and concerns of Hurricane Katrina may appear differently between Louisiana and Illinois.

Due to the inherent interaction of the content of weblogs with both time and location, it is highly desirable to analyze weblogs in a temporal and spatial context. Indeed, many interesting questions could only be answered by connecting content with time and location and analyzing spatiotemporal topic patterns. For example, a sociology researcher may ask "what do people in Florida think about the presidential candidates and how do their opinions evolve over time", while a business provider may be interested in comparing the customer responses to their new product from two countries.

Although some previous work (e.g., [77]) has considered temporal and spatial information associated with weblogs, no previous work has addressed well the need for correlating the content, especially the multiple topics *within* articles, with spatiotemporal information.

Clearly, the problem of spatiotemporal topic analysis is an instantiation of contextual text mining, or more specifically contextual topic analysis. The contexts involved in this text mining task are topics (themes), time, and geographic location. We are interested to find out how the content and the strength of topics vary over different time and location. These variations are characterized by the spatiotemporal topic patterns, an instantiation of contextual topic patterns.

5.1.2 The Problem

The general problem of spatiotemporal topic analysis can be formulated as follows.

Formally, we are given a collection of text documents with unique time stamps and location labels, $\mathcal{D} = \{d_1, ..., d_n\}$. We denote the time stamps and location labels of document d_i as $\tilde{t}_i$ and $\tilde{l}_i$, where $\tilde{t}_i \in T = \{t_1, t_2, ..., t_{|T|}\}$ and $\tilde{l}_i \in L = \{l_1, l_2, ..., l_{|L|}\}$ respectively.

To model the temporal patterns of topics, we define the concept "topic life cycle" as follows:

Definition 5.1 (Topic Life Cycle) Given a topic represented as language model θ, a location l and a set of consecutive time stamps $T = \{t_1, t_2, ..., t_{|T|}\}$, the ***topic life cycle*** of topic θ at location l is the conditional probability distribution $\{P(t|\theta, l)\}_{t \in T}$. Clearly, $\sum_{t \in T} P(t|\theta, l) = 1$. We define the ***overall life cycle*** of a topic as $\{P(t|\theta)\}_{t \in T}$ if no specific location is given.

A topic life cycle can be visualized by plotting the density function of the time-topic distribution

$\{P(t|\theta,l)\}_{t\in T}$ continuously over the entire T. Clearly, we can see that a topic life cycle is a special contextual topic pattern, which is essentially a conditional distribution of one type of context (time) given other contexts (topic and location).

Note that under this definition, the life cycles of different topics are not directly comparable. That is, given a time t, $P(t|\theta_1,l) > P(t|\theta_2,l)$ does not necessarily imply that θ_1 is stronger than θ_2 at t. To compare the strength of θ_1 and θ_2 at a given time t, we can compute $P(\theta_1|\tilde{t},l)$ and $P(\theta_2|\tilde{t},l)$ using Bayes rule.

To model spatial patterns, we further define the "topic snapshot" of the collection at a given time.

Definition 5.2 (Topic Snapshot) Given a set of topics represented as topic models $\Theta = \{\theta_1,\theta_2,...\theta_k\}$, a time stamp t and a set of locations $L = \{l_1,l_2,...,l_{|L|}\}$, the ***topic snapshot*** at time t is defined as the joint probability distribution of θ and l conditioned on t, i.e. $\{P(\theta,l|t)\}_{\theta\in\Theta,l\in L}$. Naturally, we have $\sum_{\theta\in\Theta,l\in L} P(\theta,l|t) = 1$.

A topic snapshot can be visualized by a map of topic distributions over all locations for a given time. Note that with this definition, the strength of two topics at the same location is directly comparable, i.e. given a location $\tilde{l}$, if $P(\theta_1,l|t) > P(\theta_2,l|t)$, we have that θ_1 is stronger than θ_2 at location l during the time period t. Note that a topic snapshot is also a special type of contextual topic pattern.

Intuitively, we may assume that some global topics would span the whole collection. Given a text collection $\mathcal{D}$ with time and location labels, we define the major tasks of the **Spatiotemporal Topic Analysis (STA)** problem as follows: 1) automatically extract a set of major topics from $\mathcal{D}$; 2) for a given location, compute the life cycles of the common topics at this location; 3) for a given time period, compute the topic snapshot over all locations.

Formally, given $\mathcal{D} = \{(d_1,\tilde{t}_1,\tilde{l}_1),(d_2,\tilde{t}_2,\tilde{l}_2),\ ...,\ (d_n,\tilde{t}_n,\tilde{l}_n)\}$, the task of STA is to: 1) discover topics $\Theta = \{\theta_1,...,\theta_k\}$; 2) for each given topic θ and location l, compute $P(t|\theta,l)$ for all $t \in T$; 3) for each given t, compute $P(\theta,l|t)$ for all $\theta \in \Theta$ and $l \in L$.

Next, we present how to instantiate CPLSA to solve the problem of spatiotemporal topic analysis.

5.1.3 Methods

General Idea

Previous work has shown that mixture models of multinomial distributions (i.e., mixture language models) are quite effective in extracting topics from text [61, 10, 51, 184, 113]. The basic idea of such approaches is to assume that each word in the collection is a sample from a mixture model with multiple multinomial distributions as components, each representing a topic. By fitting such a model to text data, we can obtain the distributions for the assumed topics.

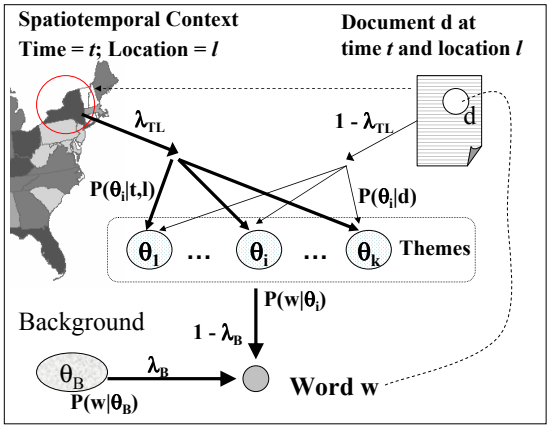


Figure 5.1: The generation process of a word in the spatiotemporal topic model

Our main idea for probabilistic spatiotemporal topic analysis is to adopt a similar approach to extract topics and extend existing work on mixture models to incorporate a time variable and a location variable. Intuitively, the words in a blog article can be classified into two categories: (1) common English words (e.g., "the", "a", "of"); (2) words related to the global subtopics (themes) whose spatiotemporal distribution we are interested in analyzing (e.g., "Hurricane Katrina and oil price"). Correspondingly, we introduce two kinds of topic models: (1) θ_B is a background topic model to capture common English words; (2) $\Theta = \{\theta_1, ..., \theta_k\}$ are k global topics to be used for all articles in the collection.

With these topic models, a document of time t and location l can be modeled as a sample of words drawn from a mixture of k global topics θ_1, ..., θ_k and a background topic θ_B. To model spatiotemporal characteristics of topics, we assume that the topic coverage in a document depends on the time and location of the document. The mixture model is illustrated in Figure 5.1.

Such a mixture model can be interpreted as modeling the following process of "writing" a weblog article: An author at time t and location l would write each word in the article by making the following decisions stochastically: (1) The author would first decide whether the word will be a non-informative English word, and if so, the word would be sampled according to θ_B. (2) If the author decided that the word should not be a non-informative word, but a content word, the author would then further decide which of the k subtopics this word should be used to describe. To make this decision, the author could use either a document-specific topic coverage distribution ($p(\theta_j|d)$) or a shared topic coverage distribution of all the articles with the same spatiotemporal context as this article ($p(\theta_j|t,l)$). (3) Suppose the j-th subtopic is picked in step (2), the remaining task is simply to sample the word according to θ_j.

We can fit such a spatiotemporal topic model to our weblog data to obtain an estimate of all the parameters in a way similar to the previous work on using mixture models for text mining. The model

parameters can then be used to compute various kinds of spatiotemporal topic patterns.

We now formally present the spatiotemporal topic model.

The Spatiotemporal Topic Model

Let $\mathcal{D} = \{d_1, ..., d_n\}$ be a weblog collection where $t_d \in T = \{t_1, t_2, ..., t_{|T|}\}$ is a time stamp and $l_d \in L = \{l_1, l_2, ..., l_{|L|}\}$ is a location label for document d. Suppose $\Theta = \{\theta_1, ..., \theta_k\}$ are k global topics. The likelihood of a word w in document d of time t and location l according to our mixture model is

$$p(w : d, t, l) = \lambda_B p(w|\theta_B) + (1 - \lambda_B) \sum_{j=1}^{k} p(w, \theta_j|d, t, l), \tag{5.1}$$

where λ_B is the probability of choosing θ_B.

We may decompose $p(w, \theta_j|d, t, l)$ in different ways to obtain interesting special cases of this general mixture model; some of them will be discussed later in Section 4.3.2. For spatiotemporal topic analysis, we decompose it as follows:

$$\begin{aligned} p(w, \theta_j|d, t, l) &= p(w|\theta_j)p(\theta_j|d, t, l) \\ &= p(w|\theta_j)((1 - \lambda_{TL})p(\theta_j|d) + \lambda_{TL}p(\theta_j|t, l)) \end{aligned} \tag{5.2}$$

where λ_{TL} is a parameter to indicate the probability of using the topic coverage distribution of the spatiotemporal context to choose a topic. $p(w|\theta_j)$ gives us the word distribution for each topic, whereas $p(\theta_j|t, l)$ gives a time and location-specific distribution of the topics, which we could exploit to compute various kinds of spatiotemporal topic patterns. The log likelihood of the whole collection C is thus

$$\log p(\mathcal{D}) = \sum_{d \in \mathcal{D}} \sum_{w \in V} c(w, d) \times \log[\sum_{j=1}^{k} p(w|\theta_j)((1 - \lambda_{TL})p(\theta_j|d) + \lambda_{TL}p(\theta_j|t_d, l_d))]. \tag{5.3}$$

where $c(w, d)$ is the count of word w in document d, t_d and l_d are the time and location labels of d.

It is easy to show that this model is a special case of the CPLSA model, or more specifically a Fixed-View CPLSA model. Indeed, since in the main tasks of spatiotemporal topic analysis, we are not interested in the content variations of topics over time and location, we assume that there is only one view in the collection - the global view. We then assume that there are many distinct coverages. For every combination of time and location, there is a corresponding topic coverage. Let us denote it as $\kappa_{t,l}$. For every document d, there is an additional topic coverage κ_d. Therefore, a word in a document can select from two topic coverages, one corresponding to the document itself (κ_d), and the other corresponding to the time and

location of the document (κ_{t_d,l_d}). We can rewrite the $p(\theta_j|d)$ and $p(\theta_j|t_d,l_d)$ in Equation 5.3 as $p(j|\kappa_d)$ and $p(j|\kappa_{t_d,l_d})$, which corresponds to the notations in Equation 4.5. We further assume that $p(\kappa_d|d) = 1-\lambda_{TL}$ and $p(\kappa_{t_d,l_d}|d) = \lambda_{TL}$ for all documents, the fixed-view CPLSA model in Equation 4.5 is exactly instantiated as Equation 5.3.

We can further incorporate a model for the background context to absorb the noise and common English words in the collection, which makes the log likelihood into

$$\begin{aligned}\log p(\mathcal{D}) &= \sum_{d\in\mathcal{D}}\sum_{w\in V} c(w,d)\times\log[\lambda_B p(w|\theta_B) \\ &+ (1-\lambda_B)\sum_{j=1}^{k} p(w|\theta_j)((1-\lambda_{TL})p(\theta_j|d)+\lambda_{TL}p(\theta_j|t_d,l_d))]. \end{aligned} \quad (5.4)$$

Parameter Estimation

In this section, we discuss how we estimate the parameters of the spatiotemporal topic model above using the maximum likelihood estimator, which chooses parameter values to maximize the data likelihood.

The general model has many parameters to estimate. However, for the purpose of spatiotemporal weblog mining, we will regularize the model by fixing some parameters. First, we set the background model as follows:

$$p(w|\theta_B) = \frac{\sum_{d\in\mathcal{D}} c(w,d)}{\sum_{w\in V}\sum_{d\in\mathcal{D}} c(w,d)}$$

Second, we set λ_B and λ_{TL} manually, which we will further discuss later in Section 5.1.4.

The parameters remaining to be estimated are thus reduced to: (1) the global topic models, $\Theta = \{\theta_1,...\theta_k\}$; (2) the document topic probabilities $p(\theta_j|d)$ where $1\le j\le k, d\in\mathcal{D}$; and (3) the spatiotemporal topic probability $p(\theta_j|t,l)$ where $1\le j\le k, t\in T, l\in L$.

We may use the Expectation-Maximization (EM) algorithm [36] to estimate all these parameters by maximizing the data likelihood. The updating formulas are shown in Figure 5.2.

In these formulas, $\{z_{d,w}\}$ is a hidden variable and $p(z_{d,w}=j)$ indicates the probability that word w in document d is generated using topic j. $\{y_{d,w,j}\}$ is another hidden variable and $p(y_{d,w,j}=1)$ indicates the probability that word w is generated using topic θ_j, and θ_j has been chosen according to the spatiotemporal topic coverage distribution ($p(\theta_j|t,l)$), as opposed to the document-specific topic distribution ($p(\theta_j|d)$).

The EM algorithm will terminate when it achieves a local maximum of the log likelihood. It may not reach the global optimal solution when there are multiple maximums. In our experiments, we use multiple trials to improve the local maximum we obtain.

The time complexity of each EM iteration is $O(knu + k|T||L||V|)$, where u is the average number of

$$
\begin{aligned}
p(z_{d,w}=j) &= \frac{(1-\lambda_B)p^{(m)}(w|\theta_j)[(1-\lambda_{TL})p^{(m)}(\theta_j|d)+\lambda_{TL}p^{(m)}(\theta_j|t_d,l_d)]}{\lambda_B p(w|B)+(1-\lambda_B)\sum_{j'=1}^{k}p^{(m)}(w|\theta_{j'})[(1-\lambda_{TL})p^{(m)}(\theta_{j'}|d)+\lambda_{TL}p^{(m)}(\theta_{j'}|t_d,l_d)]}\\
p(y_{d,w,j}=1) &= \frac{\lambda_{TL}p^{(m)}(\theta_j|t_d,l_d)}{(1-\lambda_{TL})p^{(m)}(\theta_j|d)+\lambda_{TL}p^{(m)}(\theta_j|t_d,l_d)}\\
p^{(m+1)}(\theta_j|d) &= \frac{\sum_{w\in V}c(w,d)p(z_{d,w}=j)(1-p(y_{d,w,j}=1))}{\sum_{j'=1}^{k}\sum_{w\in V}c(w,d)p(z_{d,w}=j')(1-p(y_{d,w,j'}=1))}\\
p^{(m+1)}(\theta_j|t,l) &= \frac{\sum_{d:t_d=t,l_d=l}\sum_{w\in V}c(w,d)p(z_{d,w}=j)p(y_{d,w,j}=1)}{\sum_{d:t_d=t,l_d=l}\sum_{j'=1}^{k}\sum_{w\in V}c(w,d)p(z_{d,w}=j')p(y_{d,w,j'}=1)}\\
p^{(m+1)}(w|\theta_j) &= \frac{\sum_{d\in\mathcal{D}}c(w,d)p(z_{d,w}=j)}{\sum_{w'\in V}\sum_{d\in\mathcal{D}}c(w',d)p(z_{d,w'}=j))}
\end{aligned}
$$

Figure 5.2: EM updating formulas for the spatiotemporal topic model

unique words in a document.

Extract Spatiotemporal Topic Patterns

Once we have all the parameters estimated using the EM algorithm, we may compute various kinds of spatiotemporal topic patterns using these parameters.

The topic life cycle for a given location l can be obtained by computing

$$p(t|\theta_j,l)=\frac{p(\theta_j|t,l)p(t,l)}{\sum_{t'\in T}p(\theta_j|t',l)p(t',l)}$$

where $p(t,l)$ is given by the word count in time period t at location l divided by the total word count in $\mathcal{D}$.

The topic snapshot given time stamp t can be obtained by computing

$$p(\theta_j,l|t)=\frac{p(\theta_j|t,l)p(t,l)}{\sum_{l'\in L,}\sum_{j=1}^{k}p(\theta_j|t,l')p(t,l')}$$

With the topic life cycles and topic snapshots, various spatiotemporal patterns can be discovered and analyzed. For example, topic shifting can be analyzed by plotting the life cycles of the same topic over different locations together, while topic spreading can be discovered by comparing the topic snapshots of consecutive time periods. We will show examples of such spatiotemporal pattern analysis in Section 5.1.4.

5.1.4 Experiments

We apply the proposed spatiotemporal topic model to three different data sets. For each data set, we extract a number of most salient topics and analyze their life cycles and topic snapshots. Experiments show that the proposed model performs well for different types of topics and can reveal interesting spatiotemporal patterns in weblogs.

Data Set Construction

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
Government Response	**New Orleans**	**Oil Price**	**Praying and Blessing**	**Aid and Donation**	**Personal Life**
bush 0.0716374	city 0.0633899	price 0.0772064	god 0.141807	donate 0.120228	i 0.405526
president 0.0610942	orleans 0.0540974	oil 0.0643189	pray 0.047029	relief 0.0769788	my 0.11688
federal 0.0514114	new 0.034188	gas 0.0453731	prayer 0.0417175	red 0.0702266	me 0.0601333
govern 0.0476977	louisiana 0.0234546	increase 0.0209058	love 0.0307544	cross 0.0651472	am 0.0291511
fema 0.0474692	flood 0.0227215	product 0.0202912	life 0.025797	help 0.0507348	think 0.0150206
administrate 0.0233903	evacuate 0.0211225	fuel 0.0188067	bless 0.025475	victim 0.0360877	feel 0.0123928
response 0.0208351	storm 0.01771328	company 0.0181833	lord 0.0177097	organize 0.0220194	know 0.0114889
brown 0.0199573	resident 0.0168828	energy 0.0179985	jesus 0.0162096	effort 0.0207279	something 0.00774544
blame 0.0170033	center 0.0165427	market 0.0167884	will 0.0139161	fund 0.0195033	guess 0.00748368
governor 0.0142153	rescue 0.0128347	gasoline 0.0123526	faith 0.0120621	volunteer 0.0194967	myself 0.00687533

Table 5.1: Selected topics extracted from Hurricane Katrina data set

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
Personal Life	**New Orleans**	**Government**	**Oil Price**	**Storm in Texas**	**Cause of Disaster**
i 0.378907	orleans 0.0877966	bush 0.0799938	oil 0.0929594	texas 0.0827817	warm 0.0408111
my 0.130939	new 0.0863562	president 0.0380268	price 0.0910098	storm 0.0677999	global 0.0362097
me 0.052442	city 0.0448588	govern 0.0336889	gas 0.08018	wind 0.0457836	cancer 0.0338968
am 0.0296791	levee 0.0310737	disaster 0.0327676	market 0.0274118	galveston 0.0398179	climate 0.0310257
her 0.0242092	water 0.0285081	federal 0.0291178	product 0.0247364	houston 0.0374735	change 0.0305301
feel 0.0133843	black 0.0242845	war 0.0283924	company 0.0239605	coast 0.0357728	rise 0.015924
friend 0.0114668	flood 0.0233388	iraq 0.0245395	energy 0.0235243	evacuate 0.0266514	surface 0.0123937
work 0.0102118	police 0.0175093	agent 0.0172738	cent 0.0216559	resident 0.0172511	scientist 0.0123253
love 0.00720455	superdome 0.0107274	katrina 0.0138582	barrel 0.0192692	landfall 0.0159967	temperature 0.0105363
life 0.00715712	neighborhood 0.00882865	response 0.0120409	refinery 0.0175738	tropic 0.012126	ocean 0.0102157

Table 5.2: Selected topics extracted from Hurricane Rita data set

We construct each data set by collecting blog entries that are relevant to a given topic. Note that the proposed method could be applied to any collection with time and location information.

We select three topics and construct a data set for each one by submitting a time-bounded query to Google Blog Search[1] and collecting the blog entries returned. Each entry has a pointer to the page containing its author profile. For privacy concerns, we only keep the location information. Since schema matching from different blog providers is not our focus, we only collect the blog entries from MSN Space. The basic information of each data set is presented in Table 5.3:

Data Set	# docs	Time Span(2005)	Query
Katrina	9377	08/16 - 10/04	Hurricane Katrina
Rita	1754	08/16 - 10/04	Hurricane Rita
iPod Nano	1720	09/02 - 10/26	iPod Nano

Table 5.3: Basic information of three data sets

We extract free text contents, time stamps and location labels from each document. Krovetz stemmer [75] is used to stem the text. We intentionally did not perform stop word pruning in order to test the robustness of the model. Originally, the smallest unit of a time stamp is a *day* and the smallest granularity of a location is a *city*. We group the time stamps and locations so that the data in each active unit (t, l) will not be too sparse. (Exactly how to group them will be discussed later in this section.) We then build an index for each data set with Lemur Toolkit[2], on top of which the proposed topic model is implemented.

[1] http://blogsearch.google.com
[2] http://www.lemurproject.org/

For each data set, we design our experiments as follows: (1) group the time stamps and locations into appropriate units; (2) apply the spatiotemporal model to extract a number of salient topics; (3) with the parameters estimated, compute the life cycle for each topic at each location and compute the topic snapshot for each time period; (4) visualize the results with life cycle plots and snapshot maps. The experiment details and results are discussed below.

Parameter Setting

In the spatiotemporal topic model, there are several user-input parameters which provide flexibility for the spatiotemporal topic analysis. These parameters are set empirically. In principle, it is not easy to optimize these parameters without relying on domain knowledge and information about the goal of the data analysis. However, the nature of this mining task is to provide user flexibility to explore the spatiotemporal text data with their belief about the data. We expect that the change of these parameters will not affect the major topics and trends but provide flexibility on analyzing them. The effect of the parameters is as follows.

Generally, we expect each discovered topic to be semantically coherent and distinctive from the general information of the collection, which is captured by the background model. λ_B controls the strength of the background model, and should be set based on how discriminative we would like the extracted topics to be. In practice, a larger λ_B would cause the stop words to be automatically excluded from the top probability words in each topic language model. However, an extremely large λ_B could attract too much useful information into background and make the component topic difficult to interpret. Empirically, a suitable λ_B for blog documents can be chosen between 0.9 and 0.95.

λ_{TL} controls the modeling of spatiotemporal topic distributions. A higher λ_{TL} would allow more content information of a document to be used to learn the spatiotemporal topic distribution, leaving little room for variation in individual documents. $\lambda_{TL} = 1$ would essentially pool all the documents of the same time and location and force them to use the same spatiotemporal topic distribution, whereas $\lambda_{TL} = 0$ would cause the spatiotemporal topic model to degenerate to the flat topic model. Empirically, a good selection of λ_{TL} lies between 0.5 and 0.7.

Parameter k represents the number of subtopics in a collection which can be set based on any prior knowledge about the event. When no domain knowledge is available as in our experiments, we follow [113] to determine the number of topics by enumerating multiple possible values of k and drop the topics with a significant low value of $\frac{1}{|C|}\sum_{d\in\mathcal{D}} p(\theta|d)$.

Note that the granularity of time is also a parameter which should be set carefully. A too coarse time granularity may miss interesting bursting patterns. Meanwhile, a too small granularity makes the

information in each time interval sparse, which causes the life cycle plots to be sensitive and with sharp variations. We address this problem by first selecting a reasonable small time granularity (1 day) and use a sliding window to smooth the topic life cycle at a later stage. For example, we may use $\tilde{p}(t_i|\theta, l) = \frac{1}{3}[p(t_{i-1}|\theta, l) + p(t_i|\theta, l) + p(t_{i+1}|\theta, l)]$ to substitute $p(t_i|\theta, l)$ when plotting.

In the following sections, we present interesting topics, topic life cycles and topic snapshots discovered from the three data sets.

Hurricane Katrina Data Set

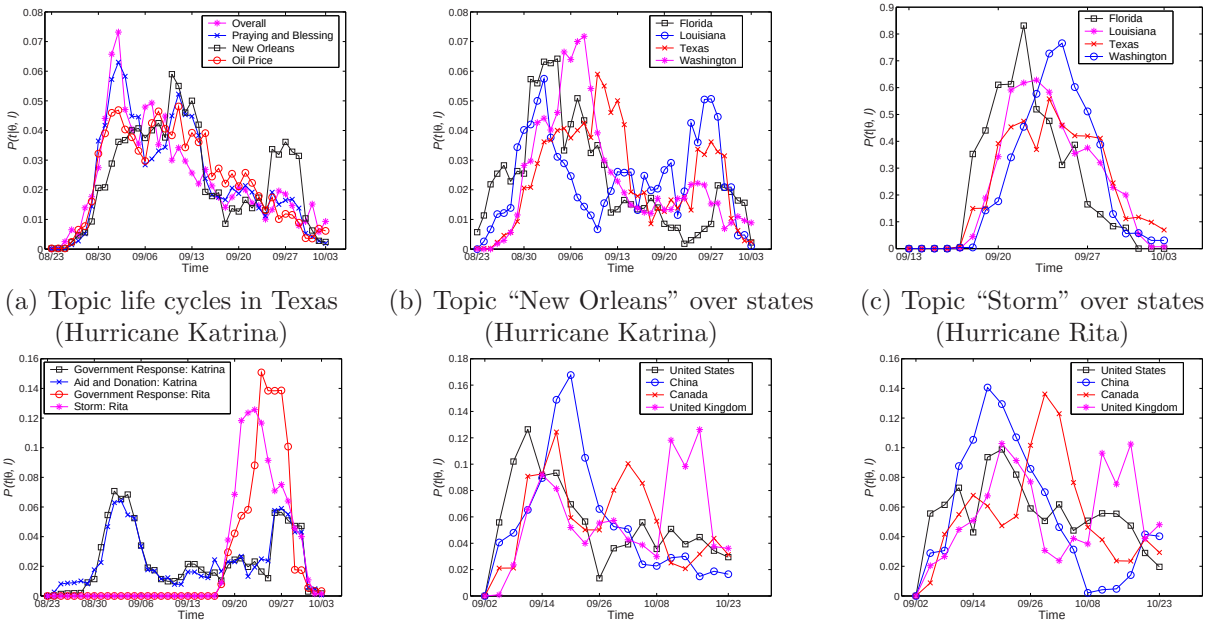
The Hurricane Katrina data set is the largest one in our experiments. 7118 documents out of 9377 have location information. We vary the time granularity from a day to a week. The extracted topics are not sensitive to this granularity change. We set the granularity of location as a *state* and analyze the topic snapshot within the United States.

The most salient topics extracted from the Hurricane Katrina data set are presented in Table 5.1, where we show the top probability words of each topic language model. The semantic labels of each topic are presented in the second row of Table 5.1. We manually label each topic with the help of the documents with highest $p(\theta|d)$. A few less meaningful topics are dropped as noise.

From Table 5.1, we can tell that topic 1 suggests the concern about "Government Response" to the disaster; topic 2 discusses the subtopic related to "New Orleans"; topic 3 represents people's concern about the increase of "Oil Price"; topic 4 is about "praying and blessing" for the victims; and topic 5 covers the aid and donations made for victims. Unlike topic 1 to topic 5, the semantics of which can be inferred from the top probability words, topic 6 is hard to interpret directly from the top words. By linking back to the original documents, we find that the documents with highest probability $p(\theta|d)$ for topic 6 tend to talk about personal life and experiences of the author. This is interesting and reasonable because weblogs are associated with personal contents. Indeed, we observe that a similar topic also occurs in other two data sets.

We then plot and compare the topic life cycles at the same location and life cycles of a topic over states. Interesting results are selectively shown in Figure 5.3(a) and (b).

Figure 5.3(a) shows the life cycles of different topics in Texas. The "Overall" life cycle shows the coverage of the overall topic measured by the collection size over time. Clearly, all topics grow rapidly during the first week, in which Hurricane Katrina was active. The topic "praying and blessing" starts dropping after ten days and increases again during the third week. In the same week, the discussion about "New Orleans" reaches the peak. Comparing with real time line of Hurricane Katrina, we find that this is the week that the mayor of New Orleans ordered evacuation. The topic "New Orleans" rises again around late September.



(a) Topic life cycles in Texas (Hurricane Katrina)

(b) Topic "New Orleans" over states (Hurricane Katrina)

(c) Topic "Storm" over states (Hurricane Rita)

(d)Topic life cycles in Louisiana (Hurricane Rita)

(e) Topic "Release of Nano" over states (iPod Nano)

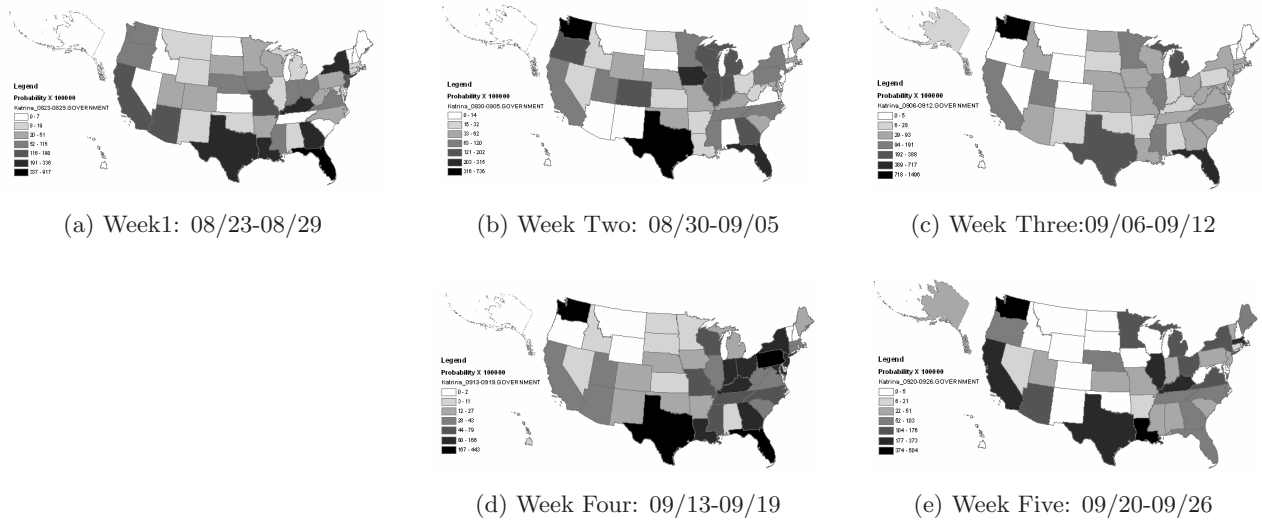
(f) Topic "Nano Features" over states (iPod Nano)

Figure 5.3: Topic life cycle patterns from three data sets

The public concern of "Oil Price", however, shows a stable high probability until the fourth week.

Figure 5.3(b) plots the life cycles of topic "New Orleans" at different states. We observe that this topic reaches the highest probability first in Florida and Louisiana, followed by Washington and Texas, consecutively. During early September, this topic drops significantly in Louisiana while still strong in other states. We suppose this is because of the evacuation in Louisiana. Surprisingly, around late September, an arising pattern can be observed in most states, which is most significant in Louisiana. Since this is the time period in which Hurricane Rita arrived, we surmise that Hurricane Rita has an impact on the discussion of Hurricane Katrina. This is reasonable since people are likely to mention the two hurricanes together or make comparisons. We can find more clues to this hypothesis from Hurricane Rita data set.

Representative snapshots for topic "Government Response" over five weeks are presented in Figure 5.4. The darker the color is, the larger the $p(\theta, l|t)$ is, but the color cannot be compared across the snapshots because $p(\theta, l|t)$ is conditioned on the time t, which differs in each snapshot. From Figure 5.4, we observe that at the first week of Hurricane Katrina, the topic "Government Response" is the strongest in the southeast states, especially those along the Gulf of Mexico. In week 2, we can clearly see the pattern that the topic is spreading towards the north and western states. This pattern continues over week 3, in which the topic is distributed much more uniformly over the States. However, in week 4, we observe that the topic converges to east states and southeast coast again. Interestingly, this week happens to overlap with the first week of



(a) Week1: 08/23-08/29 (b) Week Two: 08/30-09/05 (c) Week Three:09/06-09/12

(d) Week Four: 09/13-09/19 (e) Week Five: 09/20-09/26

Figure 5.4: Snapshots for topic "Government Response" over the first five weeks of Hurricane Katrina

Hurricane Rita, which may raise the public concern about government response again in those areas. In week 5, the topic becomes weak in most inland states and most of the remaining discussions are along the coasts. We will see comparable patterns at the same time periods from Hurricane Rita data set. Another interesting observation is that this topic is dramatically weakened in Louisiana during week 2 and 3, and becomes strong again from the fourth week. Week 2 and 3 are consistent with the time of evacuation in Louisiana.

Hurricane Rita Data Set

In the Hurricane Rita data set, 1403 documents out of 1754 have location labels. As in Hurricane Katrina, we choose a *state* as the smallest granularity of locations.

The most salient topics extracted from Hurricane Rita data set are shown in Table 5.2.

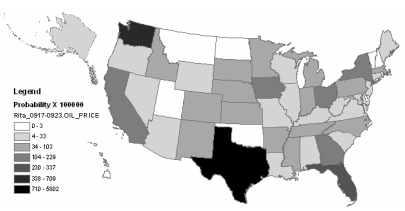
Compared with the topics extracted from the Hurricane Katrina data set, we observe that the two data sets share several similar topics. Besides the "Personal Life" topic, we see that the topic "New Orleans", "Government Response", and "Oil Price" occur in both collections. This is reasonable because the two events are comparable. The Hurricane Rita data set, however, has its specific topics such as "the Storm in Texas" and "Cause of the Disaster". We notice that the topic "Government Response" of Hurricane Rita covers extra politics contents such as Iraq War. Some topics of Hurricane Katrina, such as "Praying" and "Donation", do not appear to be salient in Hurricane Rita.

The interesting topic life cycles of Hurricane Rita are presented in Figure 5.3 (c) and (d). Figure 5.3(c) shows the life cycles of topic "Storm" over different states. Similar to Hurricane Katrina, at the very

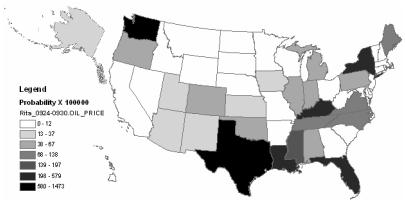
beginning, Florida is the most active state, which is also the first state where the topic reaches its peak. Shortly after Florida, the topic becomes the strongest in Louisiana, followed by Texas and Washington. In most states, the topic life cycle drops monotonically after the peak. All the life cycles fade out within two weeks from 9/17, which indicates that the impact of Hurricane Rita may not be as high as Hurricane Katrina.

Figure 5.3(d) compares the topic life cycles in Louisiana between Hurricane Katrina and Hurricane Rita. The two Hurricane Katrina topics share similar life cycle patterns and so are the two topics of Hurricane Rita. In Louisiana, the discussion of Hurricane Katrina drops rapidly around early September and rises again significantly at the last week of September. Interestingly, this arising is just shortly after the significant rising of the discussion about Hurricane Rita. This further strengthens our hypothesis that the two events have interactive impacts in weblogs. Indeed, nearly 40% blog entries which mentioned Hurricane Rita after September 26th also mentioned Hurricane Katrina.

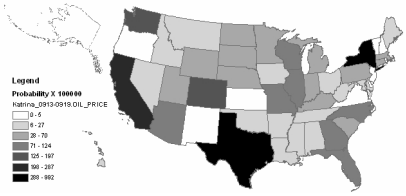
The topic snapshots over the first two weeks of Hurricane Rita show that the discussion of Hurricane Rita did not spread so significantly as the first two weeks of Hurricane Katrina. Instead, the spatial patterns are similar to the last two weeks of Hurricane Katrina, which are roughly around the same time period. We present the snapshots of the topic "Oil Price" in Figure 5.5.



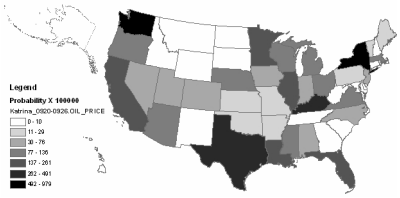
(a) Week One of Rita: 09/17/-09/23



(b) Week Two of Rita: 09/24-09/30



(c) Week Four of Katrina: 09/13-09/19



(d) Week Five of Katrina: 09/20-09/26

Figure 5.5: Snapshots for topic "Oil Price" of the first two weeks of Hurricane Rita

During the first week of Hurricane Rita, we observe that the topic "Oil Price" is already widespread over the States. In the following week, the topic does not further spread; instead, it converges back to the states strongly affected by the hurricane. Comparable patterns for the same topic can be found during the last two weeks of Hurricane Katrina. This further implies that the two comparable events have interacting impact

on the public concerns about them.

iPod Nano Data Set

This data set contains 1387 documents with location labels. We assume that the topic patterns have more significant difference between different countries rather than between states inside United States. Therefore, we discretize locations into *countries*.

Table 5.4 shows the interesting topics extracted from this data set. Again, we observe the personal life topic, which may be a characteristic topic for weblogs.

Topic 1	Topic 2	Topic 3	Topic 4
Release of Nano	**Marketing**	**Special Features**	**Personal Life**
ipod 0.2875	will 0.0306	your 0.0598	i 0.2489
nano 0.1646	market 0.0273	1 0.0478	you 0.0627
apple 0.0813	search 0.0270	music 0.0455	have 0.0312
september 0.0510	apple 0.0257	song 0.0378	my 0.0269
mini 0.0442	company 0.0215	display 0.0209	am 0.0220
screen 0.0242	itune 0.0200	shrink 0.0182	know 0.0214
new 0.0200	phone 0.0199	4gb 0.0130	want 0.0148
mp3 0.0155	web 0.0186	color 0.0102	thing 0.0146
thin 0.0140	microsoft 0.0185	pencil-thin 0.0100	would 0.0130
shuffle 0.0127	motorola 0.0151	model 0.0096	think 0.0110

Table 5.4: Selected topics from iPod Nano data set

In the rest three topics, topic 1 is about the news that iPod Nano was introduced. Topic 2 is about the marketing of Apple and how it is related to other business providers. Topic 3 is about specific features of iPod Nano.

We compare the life cycles of topics over different countries. Our expectation is that the life cycle of topics in the United States would evolve faster than those outside. The results are shown in Figure 5.3 (e) and (f).

For the topic about the release of iPod Nano, United States is indeed the first country where it reaches the top of its life cycle, followed by Canada, China, and United Kingdom consecutively. The topic in China presents a sharp growing and dropping, which indicates that most discussions there are within a short time period. The life cycles in Canada and United Kingdom both have two peaks.

Similar patterns can be found in the topic discussing the specific features of Nano. All life cycles start around early September. At the very beginning, discussions in the United States surge more quickly than in any other countries. The topic reaches its peak in Canada posteriorly to the other countries. There are also two peaks in United Kingdom.

To summarize, the experiments on three different data sets show that the spatiotemporal topic model we proposed in Section 5.1.3 can extract topics and their spatiotemporal patterns effectively. The comparative analysis of topic life cycles and topic snapshots is potentially useful to reveal interesting patterns and to answer a lot of questions.

5.2 Temporal-Author-Topic Analysis

In the previous section, we show that the fixed-view version of CPLSA is effective in a real world text mining application - spatiotemporal topic analysis. In this section and the next one, we briefly introduce two other experiments on two different applications of contextual text mining, to further illustrate the effectiveness of CPLSA and its instantiations.

In this experiment, we evaluate the performance of the CPLSA models on author-topic comparative analysis. If two authors have similar research interest, we assume that there is a set of common topics which can be found in their publications. Since different author has different preferences and focuses, the content of these topics will also vary corresponding to each author. Previous work on author-topic analysis only consider the authorship of documents as the context [162]. Intuitively, however, the topics that an author favors also evolute over time. We add another type of context information, i.e., publication time, to test the effectiveness of our model on handling multiple types of contexts.

We collect the abstracts of 282 papers published by two famous Data Mining researchers from ACM Digital library. We split the whole time line into three spans: before the year 1993, from 1993 to 1999, and after the year 1999. This will give us 12 possible views as in Table 5.5. Since we are not interested in analyzing the coverage variations across contexts (i.e. time and authors), we assume the coverage of topics only depends on documents but not on the contexts.

#Context Features	Views (A and B are two authors)
0	Global View
1	A; B; < 92; $93 \sim 99$; $00 \sim 05$
2	A, < 92; A, $93 \sim 99$; A, $00 \sim 05$ B, < 92; B, $93 \sim 99$; B, $00 \sim 05$

Table 5.5: Possible Views in Author-Topic Analysis

Schedulers					
Global	Author A	Author B	Author A: 2000∼	1993∼1999	2000∼
pattern 0.110689	project 0.0444375	research 0.0550772	close 0.0805878	rule 0.0616733	index 0.0430914
frequent 0.040613	itemset 0.0432976	next 0.0308254	pattern 0.072078	distribute 0.0567852	graph 0.0343051
frequent-pattern 0.0393	intertransaction 0.03072	transition 0.0308254	sequential 0.0462879	researcher 0.0324659	web 0.0306886
sequential 0.0359059	support 0.0264818	panel 0.0275384	min_support 0.03526	algorithm 0.0217309	gspan 0.0273849
method 0.0214187	associate 0.0258175	technical 0.0275384	length 0.0315721	over 0.0162951	substructure 0.02005
pattern-growth 0.02035	frequent 0.0181942	technology 0.0258949	threshold 0.0296533	fdm 0.0227141	gindex 0.016431
condense 0.0184008	closet 0.0176081	article 0.0154127	frequent 0.0196054	study 0.0116576	bide 0.016431
increment 0.0138457	apriori 0.0170468	revolution 0.0154127	top-k 0.0176324	scable 0011357.	magnitude 0.0151909
constraint 0.0130636	prefixspan 0.0130272	tremendous 0.0154127	without 0.0175662	pass 0.011357	size 0.0114699
push 0.0103159	pseudo 0.0109016	innovate 0.0154127	fp-tree 0.0102471	disclose 0.011357	xml 0.010954

Table 5.6: Comparison of the content of topic "Frequent Pattern Mining" over different views

Therefore, we use the FC-CPLSA model presented in Section 4.3.2 to model the topics and their views corresponding to different contexts. Our goal is thus to estimate all the parameters in the regularized model, and compare $p(w|\theta_{jl})$ over different view v_j.

To avoid the EM algorithm being trapped in suboptimal local maximums, we need to make associations

between each θ_{jl} to its corresponding global view θ_l. We achieve this by selecting a good starting point for the EM algorithm. Specifically, we begin with a prior of a large $p(v_0|d)$ to view 0, which is the global view. This ensures us to get the strong signal of global topics instead of local biased topics. In the following iterations, we gradually decay this prior and terminate the EM algorithm early when the average view distribution for view 0 (i.e., $\sum_{d\in\mathcal{D}} p(v_0|d)/|\mathcal{D}|$) drops under a threshold, say 0.1. This gives us a good starting point for the EM algorithm. Then, we do this procedure again for multiple trials and select the best start point (i.e., the one with the highest likelihood). Finally, we run the EM algorithm beginning with this selected start point until it converges. The results for this experiment are selectively presented in the following table.

In Table 5.6, we see that the content of this selected topic varies over different views. From the global view, in which all documents are included, we can tell that this topic is talking about frequent pattern mining. From the view of Author A, we see specific frequent pattern mining techniques such as database projection, apriori, prefixspan, and closet. From the view of Author B, we see that he is not as deep into techniques of mining frequent patterns, but rather more associated with introductional and innovated work of frequent pattern mining. From the view of the years before 1993, the corresponding topic barely has any connection to frequent pattern mining. This is reasonable however, since the first and most influential paper of frequent pattern mining was published in 1993. From the view of year 1993 to 1999, we see that this topic evolutes to talk about association rules, which is perhaps the most important application of frequent pattern mining at that time. Specific techniques, such as fdm (Fast Distributed Mining of associate rules) appears high in the word distribution. From the view of the years after 1999, it is interesting to see the appearance of more new applications of frequent pattern mining, such as graphs and web. The terms corresponding to specific techniques of mining graph patterns and sequential patterns, e.g., gspan and bide, are with high probabilities in the topic word distribution. In the view corresponding to a combined context (Author A and after 1999), the top terms include “close”, “top-k”, and “fp-tree”, which well reveal the preferences of author A in frequent pattern mining. The view specific topic for the combined context “Author B after 1999” is not well associated with the global topic again, which is consistent to the fact that Author B is not activate in frequent pattern mining any more after 2000.

This experiment shows that the CPLSA model can extract and compare the topic variations over different views effectively.

5.3 Event Impact Analysis

In many scenarios, a collection of documents are usually associated with a series of events. For example, weblogs usually reflect people's opinions about the events happening. The research topics covered by scientific literatures are also likely to be affected by the influential related events, such as the invention of WWW, and the proposing of a new research direction. The impact of such event can usually be analyzed by comparing the topics in the documents published before versus after the event. In this experiment, we apply CPLSA directly on the problem of event impact analysis. Since each event gives a possible segmentation of the time line, this analysis also provides an evaluation of CPLSA on modeling overlapping views that are not orthogonal to each other. Although the experiments in previous sections also covers some overlapping views (e.g., a view corresponding to a location and a view corresponding to a time stamp), these overlaps are caused by different types of, or orthogonal context features (e.g., time and location). In reality however, the overlapping views with the same type of context feature is desirable. For example, a business analyzer may need to analyze and compare the customers' opinions in the first week, in the first month, in the first season, or in the first year after a new product is released. One strength of our model is that we allow the analysis views that overlap with each other. In this experiment, we evaluate our model on event impact analysis and overlapping view analysis.

We collect the abstracts of 1472 papers published in 28 years' SIGIR conferences from ACM Digital Library. We select two influential events to the Information Retrieval community in the 90s. One is the beginning of Text REtrieval Conferences (TREC) in 1992, which provide large-scale standard text datasets and judgements for many retrieval problems. The other is the introduction of language model into Information Retrieval in 1998, which began a genre of research and led to a lot of publications. Our goal is to use the CPLSA model to reveal the impact of these two events in IR research, i.e., how the content of research topics change after the two events.

To achieve this, we assign the abstracts in SIGIR proceedings into four contexts, each corresponds to a time span. The first context includes all the documents were published before 1993, in which is the first SIGIR conference after the start of TREC. The second context contains documents published on or after that. The third context includes abstracts before the year 1998, in which the first paper of language model in information retrieval was published. The fourth context contains all abstracts published on or after 1998. It is clear that there are overlaps between these contexts. We also include a global view, which corresponds to all the abstracts in SIGIR proceedings.

We use the same strategy as presented in Section 5.2 to avoid the EM algorithm to be trapped in unexpected local maximums. We extract 10 salient global topics from this collection and present the most

interesting one.

Views:	Global	Pre-Trec	Post-Trec	Pre-Language Model	Post-Language Model
SIGIR	term 0.159983	vector 0.0514067	xml 0.0677684	probabilist 0.0777954	model 0.16867
	relevance 0.0751814	concept 0.0297583	element 0.0212121	model 0.0431573	language 0.0752643
	weight 0.0659849	extend 0.0297405	email 0.0197383	logic 0.0403557	estimate 0.0520434
	feedback 0.0372254	model 0.0291697	collect 0.0191258	ir 0.0337741	parameter 0.0281169
	independence 0.031063	space 0.0236088	locate 0.0187425	boolean 0.028073	distribution 0.0268227
	model 0.0309212	boolean 0.0151455	judgment 0.0140086	fuzzy 0.0201544	probable 0.0205655
	frequent 0.0233021	function 0.0123171	rank 0.010205	algebra 0.0199632	smooth 0.0197662
	probabilist 0.018762	u 0.00898533	overlap 0.00975133	probable 0.0124902	score 0.0166799
	document 0.0173198	feedback 0.00860945	contextual 0.00936265	estimate 0.0119202	retrieval 0.0137085
	assume 0.0172082	specify 0.0083182	solution 0.00913	weight 0.0111257	markov 0.0118979
	dependency 0.0157547	correlate 0.00779721	subtopic 0.00791172	rank 0.0107045	likelihood 0.00585364

Table 5.7: Comparison of topic content over different views in SIGIR collection

From the global view in Table 5.7, we see that this topic is talking about retrieval models, especially term weighting and relevance feedback. The content of this common topic varies from different views. From the Pre-Trec view, which corresponds to the time before 1993, we see that vector space model dominates, and boolean queries are mentioned frequently. In the Post-Trec view, however, we notice that XML retrieval model has been paid more attention to. Also, we see specific types of data (email) and other terms related to the nature of TREC (e.g., collect, judgement, rank). It is more interesting when comparing the view "Pre-Language Model" and "Post-Language Model". We see that before 1998, the retrieval models are dominated by probabilistic models. After 1998, however, it is very clear that language model dominates the topic. The top ranked terms have changed to indicate language models, parameter estimations, likelihood and probability distributions, and language model smoothing. This is consistent with our prior knowledge. The overlapping views, for example Pre-LM and Pre-Trec, do share some content but clearly with different focuses. Pre-Trec, which is more faraway, emphasizes vector space model while Pre-LM emphasizes probabilistic models. This experiment shows that our method is effective to analyze event impact and model the overlapping views.

5.4 Related Work

The aim of this chapter is to present the effectiveness of the CPLSA model and its special versions on a variety of real world text mining tasks.

The most relevant work to CPLSA is the Probabilistic Latent Semantic Analysis model (PLSA) proposed by Hofmann [59, 61]. Our CPLSA model is a natural extension of PLSA to incorporate context other than topics. To avoid overfitting in PLSA, Blei and co-authors proposed a generative aspect model called Latent Dirichlet Allocation (LDA), which could also extract a set of topics from a document collection [10]. The same contextual extension can be expected to apply on LDA.

Some recent work of topic modeling have considered some specific types of context. For example, temporal context is considered in [51, 137, 20, 113]. Multi-collection context is analyzed in [184]. Author-topic

analysis is proposed in [162]. Li et al. proposed a probabilistic model to detect retrospective news events by explaining the generation of "four Ws[3]" from each news article [91]. Our work is a generalization of these studies of specific context and provides a general probabilistic model which can incorporate all kinds of context with topic context.

Besides the general CPLSA model, the particular text mining applications we select is related to the following work.

Spatiotemporal Text Mining

Temporal context is also addressed in Kleinberg's work on discovering bursty and hierarchical structures in streams [72] and some work on topic/event/trend detection and tracking (e.g., [1, 13, 99, 74, 126]). However, most of this work assumes one document only belongs to one topic and cannot be easily generalized to analyze other contexts.

To the best of our knowledge, the problem of spatiotemporal text mining has not been well studied in existing work.

Most existing text mining work (e.g., [124, 113]) does not consider the temporal and location context of text. Li and others proposed a probabilistic model to detect retrospective news events by explaining the generation of "four Ws[4]" from each news article [91]. However, their work considers time and location as independent variables, and aims at discovering the reoccurring peaks of events rather than extract the spatiotemporal patterns of topics.

[51] presents a generative model similar to the one in [10] to extract scientific topics from PNAS abstracts and a post hoc analysis on the popularity of topics to detect the hot and cold topics. Our work differs from theirs in that we model the temporal dynamics of topics simultaneously with topic extraction in the statistical model. Another related work to topic life cycle analysis is [137], where a Multinomial PCA model is used to extract topics from text and analyze temporal trends. However, none of this work models the spatial information of a text collection.

Spatiotemporal data mining on numerical data and moving objects has been well studied [41, 101]. [128] present a spatiotemporal clustering method to detect the emerging space-time clusters. However, these techniques aim at analyzing explicit data objects, which cannot be used for extracting and analyzing latent topic patterns from a text collection.

[3] who, when, where and what (keywords)
[4] who (persons), when (time), where (locations) and what (keywords)

Weblog Analysis

Another line of research related to our work is weblog analysis and mining. Existing work has explored either structural analysis on communities [167, 76] and temporal analysis on blog contents [50, 53]. Our work differs from the existing work in two aspects: (1) we model the multiple topics within each blog article; (2) we correlate the contents, location and time of articles in a unified probabilistic model. None of these has been done in the previous work of weblog analysis.

Kumer and others showed that the structure and interest clusters on blogspace are highly correlated to the locality property of weblogs [77]. Although this work considered temporal and spatial *distribution* of weblogs, neither this work nor any other previous work has addressed the *content* analysis with spatiotemporal information. We consider this work as an important evidence that spatial analysis on weblog content is desired.

Some existing work further explored content and structure evolutions of weblogs for higher level tasks. For example, Gruhl and others' work in 2004 modeled information diffusion through blogspage by categorizing temporal topic patterns into spikes and chatter [53]. Their following work in 2005 explored the spike patterns of discussion of books to predict spikes in their sales rank [52].

The general spatiotemporal topic analysis methods proposed in our work can provide fundamental utilities to facilitate such higher-level predictions.

Author-Topic Analysis

Author-topic analysis is first introduced in [162], which extends LDA to model the association between authors and topics. [159] extends this work to model the senders and receivers in a collection of emails. Unlike CPLSA, this work cannot be generalized to handle contexts other than the author. In the Temporal-Author-Topic analysis we presented, a new time context is incorporated with the context of author and topics.

Event-Impact Analysis

Topic detection and tracking [13, 144, 99] aims at detecting emerging new topics and identifying boundaries of existing events. However, most of those works focus on the detection of "events" rather than summarizing the impact of the events.

5.5 Summary

In this chapter, we presented the effectiveness of the general contextual topic model (i.e., CPLSA) in real world text mining applications. Three real applications are presented, including spatiotemporal topic analysis in weblogs, temporal-author-topic analysis, and event-impact analysis in scientific literature.

Although these tasks are quite different, they share the same characteristics: all of them instantiations of the general contextual text mining problem, more specifically instantiations of the contextual topic analysis problem. All the tasks involve topic and additional explicit contexts.

With empirical experiments on real world data, we see that the CPLSA model and its two special versions - Fixed-View CPLSA and Fixed-Coverage CPLSA are quite effective in these real world tasks. The findings have proved the effectiveness of the functional framework - contextual topic analysis, and the contextual topic model - CPLSA in Chapter 4.

Besides proving the effectiveness of the CPLSA model, the solution of the three applications also makes significant contribution to text mining.

Weblogs usually have a mixture of subtopics and exhibit spatiotemporal content patterns. Discovering topics and modeling their spatiotemporal patterns are beneficial not only for weblog analysis, but also for many other applications and domains. The results show that our method can effectively discover major interesting topics from text and model their spatiotemporal distributions and evolutions. The mining results can be used to further support higher level analysis tasks such as user behavior prediction, information diffusion and blogspace evolution analysis.

The proposed spatiotemporal topic model is completely unsupervised and can be applied to any text collection with time stamps and location labels, such as news articles and customer reviews. The method thus has many potential applications, such as **(1) Search result summarization:** Provide a summary for blogsearch results, which consists of topics, snapshots of spatial distributions of topics, and temporal evolution patterns of topics. **(2) Public opinion monitoring:** Extract the major public concerns for a given event, compare the spatial distributions of these concerns, and monitor their changes over time. **(3) Web analysis:** Extract major topics and model the macro-level information spreading and evolution patterns on the blogspace. **(4) Business intelligence:** Facilitate the discovery of customer opinions/concerns and the analysis of their spatial distributions and temporal evolutions.

Besides the spatiotemporal topic analysis in weblogs, the temporal-author-topic analysis and the event-impact analysis also provides us a novel and effective way to digest the topics and contextual patterns of topics in scientific literature.

Chapter 6

Contextual Topic Analysis with Implicit Context

In the previous chapter, we introduced instantiations of the contextual topic analysis framework on three real world text mining applications. The commonness of all the three applications is that they involve topics and additional explicit context(s). In some real world applications, we encounter both topics and additional types of implicit contexts, such as sentiments. In this chapter, we study another instantiation of contextual topic analysis with the focus on implicit contexts. Specifically, we study the contextual topic model with sentiments as the additional context besides topics. We introduce another instantiation of the CPLSA model, called topic-sentiment mixture, which handles a mixture of implicit contexts including topics and sentiments (see [110]). We further elaborate on how to add priors into the contextual topic model, so that our prior understanding about the implicit contexts (including topics and sentiments) can be incorporated as a guidance to the model. Based on the topic-sentiment mixture model and the method to guide the model with prior knowledge, we introduce an interesting contextual topic analysis task to generate multi-faceted opinion summary. This chapter gives a detailed discussion about a general component of contextual topic analysis - adding model priors into contextual topic models (Section 4.4), and provides us a general guideline to model implicit contexts in contextual text mining.

6.1 Overview

More and more internet users now publish online dairies and express their opinions with Weblogs (i.e., blogs). The wide coverage of topics, dynamics of discussion, and abundance of opinions in Weblogs make blog data extremely valuable for mining user opinions about all kinds of topics (e.g., products, political figures, etc.), which in turn would enable a wide range of applications, such as opinion search for ordinary users, opinion tracking for business intelligence, and user behavior prediction for targeted advertising.

Technically, the task of mining user opinions from Weblogs boils down to sentiment analysis of blog data – identifying and extracting positive and negative opinions from blog articles. Although much work has been done recently on blog mining [76, 53, 52, 111], most existing work aims at extracting and analyzing topical

contents of blog articles without any analysis of sentiments in an article. The lack of sentiment analysis in such work often limits the effectiveness of the mining results. For example, in [52], a burst of blog mentions about a book has been shown to be correlated with a spike of sales of the book in Amazon.com. However, a burst of *criticism* of a book is unlikely to indicate a growth of the book sales. Similarly, a decrease of blog mentions about a product might actually be caused by the decrease of *complaints* about its defects. Thus understanding the positive and negative opinions about each topic/subtopic of the product is critical to making more accurate predictions and decisions.

There has also been some work trying to capture the positive and negative sentiments in Weblogs. For example, Opinmind [130] is a commercial weblog search engine which can categorize the search results into positive and negative opinions. Mishne and others analyze the sentiments [122] and moods [123] in Weblogs, and use the temporal patterns of sentiments to predict the book sales as opposed to simple blog mentions. However, a common deficiency of all this work is that the proposed approaches extract only the overall sentiment of a query or a blog article, but can neither distinguish different subtopics within a blog article, nor analyze the sentiment of a subtopic. Since a blog article often covers a mixture of subtopics and may hold different opinions for different subtopics, it would be more useful to analyze sentiments at the level of subtopics. For example, a user may like the *price* and *fuel efficiency* of a new Toyota Camry, but dislike its *power* and *safety* aspects. Indeed, people tend to have different opinions about different features of a product [184, 96]. As another example, a voter may agree with some points made by a presidential candidate, but disagree with some others. In reality, a general statement of good or bad about a query is not so informative to the user, who usually wants to drill down in different facets and explore more detailed information (e.g., "price", "battery life", "warranty" of a laptop). In all these scenarios, a more in-depth analysis of sentiments in specific aspects of a topic would be much more useful than the analysis of the overall sentiment of a blog article.

To improve the accuracy and utility of opinion mining from blog data, we propose to conduct an in-depth analysis of blog articles to reveal the major topics in an article, associate each topic with sentiment polarities, and model the dynamics of each topic and its corresponding sentiments. Such topic-sentiment analysis can potentially support many applications. For example, it can be used to generate a more detailed topic-sentiment summary of Weblog search results as shown in Figure 6.1.

In Figure 6.1, given a query word representing a user's ad hoc information need (e.g., a product), the system extracts the latent facets (subtopics) in the search results, and associates each subtopic with positive and negative sentiments. From the example sentences on the left, which are organized in a two dimensional structure, the user can understand the pros and cons of each facet of the product, or what are its best and

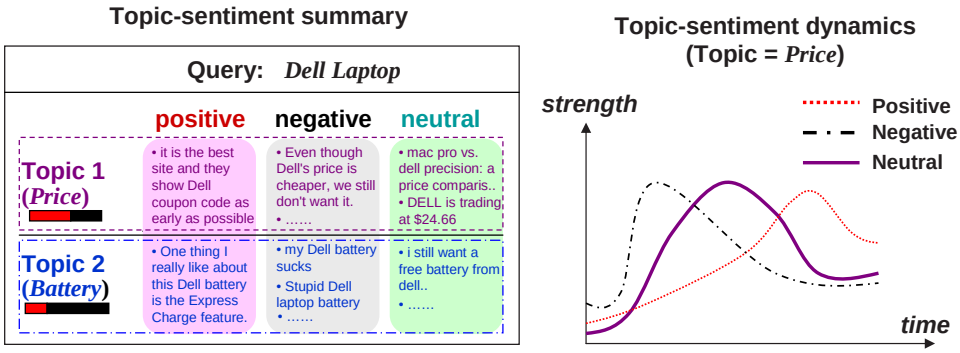


Figure 6.1: A possible application of topic-sentiment analysis

worst aspects. From the strength dynamics of a topic and its associated sentiments on the right, the user can get deeper understanding of how the opinions about a specific facet change over time. To the best of our knowledge, no existing work could simultaneously extract multiple topics and different sentiments from Weblog articles.

In this chapter, we study the novel problem of modeling subtopics and sentiments simultaneously in Weblogs. We formally define the **Topic-Sentiment Analysis** (TSA) problem and propose a probabilistic mixture model called **Topic-Sentiment Mixture** (TSM) to model and extract the multiple subtopics and sentiments in a collection of blog articles. Specifically, a blog article is assumed to be "generated" by sampling words from a mixture model involving a background language model, a set of topic language models, and two (*positive* and *negative*) sentiment language models. With this model, we can extract the topic/subtopics from blog articles, reveal the correlation of these topics and different sentiments, and further model the dynamics of each topic and its associated sentiments. We evaluate our approach on different weblog data sets. The results show that our method is effective for all the tasks of the topic-sentiment analysis.

The proposed approach is quite general and has many potential applications. The mining results are quite useful for summarizing search results, monitoring public opinions, predicting user behaviors, and making business decisions. Our method requires no prior knowledge about a domain, and can extract general sentiment models applicable to any ad hoc queries. Although we only tested the TSM on Weblog articles, it is applicable to any text data with mixed topics and sentiments, such as customer reviews and emails.

The rest of the chapter is organized as follows. In Section 6.2, we formally define the problem of Topic-Sentiment Analysis. In Section 6.3, we present the Topic-Sentiment Mixture model and discuss the estimation of its parameters. We show how to extract the dynamics of topics and sentiments in Section 6.4, and present our experiment results in Section 6.5. In Sections 6.6 and 6.7, we discuss the related work and conclude.

6.2 Problem Formulation

In this section, we formally define the general problem of Topic-Sentiment Analysis.

Let $\mathcal{D} = \{d_1, d_2, ..., d_m\}$ be a set of documents (e.g., blog articles). We assume that $\mathcal{D}$ covers a number of topics, $\{\theta_1, \theta_2, ..., \theta_k\}$. Following [134, 132, 96], we assume that there are two sentiment polarities in Weblog articles, the *positive* and the *negative* sentiment. The two sentiments are associated with each topic in a document, representing the positive and negative opinions about the topic.

Definition 6.1 (Sentiment Model) A ***sentiment model*** in a text collection $\mathcal{D}$ is a probabilistic distribution of words representing either *positive* opinions ($\{p(w|\theta_P)\}_{w \in V}$) or *negative* opinions ($\{p(w|\theta_N)\}_{w \in V}$). We have $\sum_{w \in V} p(w|\theta_P) = 1$ and $\sum_{w \in V} p(w|\theta_N) = 1$.

Sentiment models are orthogonal to topic models in the sense that they would assign high probabilities to general words that are frequently used to express sentiment polarities whereas topical models would assign high probabilities to words representing topical contents with neutral opinions.

Definition 6.2 (Sentiment Coverage) A ***sentiment coverage*** of a topic in a document (or a collection of documents) is the relative coverage of the neurtral, positive, and negative opinions about the topic in the document (or the collection of documents). Formally, we define a sentiment coverage of topic θ_i in document d as $c_{i,d} = \{\delta_{i,d,F}, \delta_{i,d,P}, \delta_{i,d,N}\}$. $\delta_{i,d,F}$, $\delta_{i,d,P}$,$\delta_{i,d,N}$ are the coverage of neutral, positive, and negative opinions, respectively; they form a probability distribution and satisfy $\delta_{i,d,F} + \delta_{i,d,P} + \delta_{i,d,N} = 1$.

In many applications, we also want to know how the neutral discussions, the positive opinions, and the negative opinions about the topic (subtopic) change over time. For this purpose, we introduce two additional concepts, "topic life cycle" and "sentiment dynamics" as follows.

Definition 6.3 (Topic Life Cycle) A ***topic life cycle***, also known as a *theme life cycle* in [113], is a time series representing the strength distribution of the neutral contents of a topic over the time line. The strength can be measured based on either the amount of text which a topic can explain [113] or the relative strength of topics in a time period [111, 114]. In this chapter, we follow [113] and model the topic life cycles with the amount of document content that is generated with each topic model in different time periods.

Definition 6.4 (Sentiment Dynamics) The ***sentiment dynamics*** for a topic θ is a time series representing the strength distribution of a sentiment $s \in \{P, N\}$ associated with θ. The strength can indicate how much positive/negative opinion there is about the given topic in each time period. Being consistent with topic life cycles, we model the sentiment dynamics with the amount of text associated with topic θ that is generated with each sentiment model.

Based on the concepts above, we define the major tasks of **Topic-Sentiment Analysis** (**TSA**) on weblogs as: **(1) Learning General Sentiment Models:** Learn a sentiment model for positive opinions and

a sentiment model for negative opinions, which are general enough to be used in new unlabeled collections. **(2) Extracting Topic Models and Sentiment Coverages:** Given a collection of Weblog articles and the general sentiment models learnt, customize the sentiment models to this collection, extract the topic models, and extract the sentiment coverages. **(3) Modeling Topic Life Cycle and Sentiment Dynamics:** Model the life cycles of each topic and the dynamics of each sentiment associated with that topic in the given collection.

It is not hard to show that the problem of topic-sentiment analysis is an instantiation of contextual text mining, more specifically contextual topic analysis. Two types of context are involved - topics and sentiments. Both of them are implicit contexts, the domain of which is uncertain. The target of the tasks, sentiment models, topic models, sentiment coverages, topic life cycles, and sentiment dynamics are all special cases of the contextual patterns, which are either represented by the conditional distributions involving contexts or can be refined from those conditional distributions. The goal is to model the implicit context, sentiment, in the contextual topic model and extract the aforementioned contextual patterns.

This problem as defined above is more challenging than many existing topic extraction tasks and sentiment classification tasks for several reasons. First, it is not immediately clear how to model topics and sentiments simultaneously with a mixture model. No existing topic extraction work [61, 10, 113, 111, 114] could extract sentiment models from text, while no sentiment classification algorithm could model a mixture of topics simultaneously. Second, it is unclear how to obtain sentiment models that are independent of specific contents of topics and can be generally applicable to any collection representing a user's ad hoc information need. Most existing sentiment classification methods overfit to the specific training data provided. Finally, computing and distinguishing topic life cycles and sentiment dynamics is also a challenging task. In the next section, we will present a unified probabilistic approach to solve these challenges.

6.3 A Mixture Model for Theme and Sentiment Analysis

6.3.1 The Generation Process

A lot of previous work has shown the effectiveness of mixture of multinomial distributions (mixture language models) in extracting topics (themes, subtopics) from either plain text collections or contextualized collections [61, 10, 113, 111, 114, 90]. However, none of this work models topics and sentiments simultaneously; if we apply an existing topic model on the weblog articles directly, none of the topics extracted with this model could capture the positive or negative sentiment well.

To model both topics and sentiments, we also use a mixture of multinomials, but extend the model

structure to include two sentiment models to naturally capture sentiments.

In the previous work [111, 114], the words in a blog article are classified into two categories: (1) common English words (e.g., "the", "a", "of") and (2) words related to a topical theme (e.g., "nano", "price", "mini" in the documents about iPod). The common English words are captured with a background component model [184, 113, 111], and the topical words are captured with topic models. In our topic-sentiment model, we extend the categories for the topical words in existing approaches. Specifically, for the words related to a topic, we further categorize them into three sub-categories: (1) words about the topic with neutral opinions (e.g., "nano", "price"); (2) words representing the positive opinions of the topic (e.g., "awesome", "love"); and (3) words representing the negative opinions about the topic (e.g., "hate", "bad"). Correspondingly, we introduce four multinomial distributions: (1) θ_B is a background topic model to capture common English words; (2) $\Theta = \{\theta_1, ..., \theta_k\}$ are k topic models to capture neutral descriptions about k global subtopics in the collection; (3) θ_P is a positive sentiment model to capture positive opinions; and (4) θ_N is a negative sentiment model to capture negative opinions for all the topics in the collection.

According to this mixture model, an author would "write" a Weblog article by making the following decisions stochastically and sampling each word from the component models: (1) The author would first decide whether the word will be a common English word. If so, the word would be sampled according to θ_B. (2) If not, the author would then decide which of the k subtopics the word should be used to describe. (3) Once the author decides which topic the word is about, the author will further decide whether the word is used to describe the topic neutrally, positively, or negatively. (4) Let the topic picked in step (2) be the j-th topic θ_j. The author would finally sample a word using θ_j, θ_P or θ_N, according to the decision in step(3). This generation process is illustrated in Figure 6.2.

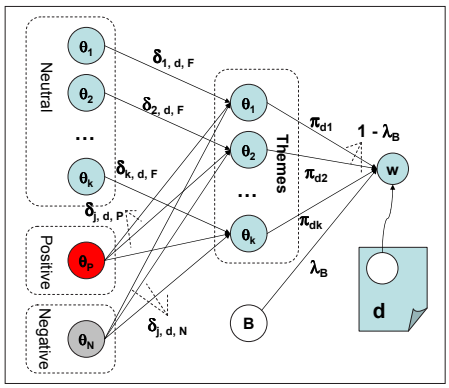


Figure 6.2: The generation process of the topic-sentiment mixture model

We now formally present the Topic-Sentiment Mixture model and the estimation of parameters based on

blog data.

6.3.2 The Topic-Sentiment Mixture Model

Let $\mathcal{D} = \{d_1, ..., d_m\}$ be a collection of weblog articles, $\Theta = \{\theta_1, ..., \theta_k\}$ be k topic models, θ_P and θ_N be a positive and negative sentiment model respectively. The log likelihood of the whole collection $\mathcal{D}$ according to the TSM model is

$$\begin{aligned} log(\mathcal{D}) &= \sum_{d \in \mathcal{D}} \sum_{w \in V} c(w,d) log[\lambda_B p(w|B) + (1-\lambda_B) \sum_{j=1}^{k} \pi_{dj} \times \\ &\quad (\delta_{j,d,F} p(w|\theta_j) + \delta_{j,d,P} p(w|\theta_P) + \delta_{j,d,N} p(w|\theta_N))]. \end{aligned} \tag{6.1}$$

where $c(w,d)$ is the count of word w in document d, λ_B is the probability of choosing θ_B, π_{dj} is the probability of choosing the j-th topic in document d, and $\{\delta_{j,d,F}, \delta_{j,d,P}, \delta_{j,d,N}\}$ is the sentiment coverage of topic j in document d, as defined in Section 6.2.

Although this model is novel, we can show that it is also a special case of the CPLSA model discussed in Chapter 4 with a small relaxation. Indeed, in the model shown in Equation 6.1, let us define 4 views. One view corresponds to the positive sentiment (v_p), one view corresponds to the negative sentiment (v_n), one view corresponds to the neutral (fact) sentiment (v_f), and an additional view corresponds to the background (v_b). We then constraint the topic models in these views, such that for every topic j, $\theta_{pj} = \theta_P$, $\theta_{nj} = \theta_N$, $\theta_{fj} = \theta_j$, and $\theta_{bj} = \theta_B$. The view selection probabilities for each document d are constricted as $p(v_b|d) = \lambda_B, p(v_f|d) = (1-\lambda_B)\delta_{j,d,F}, p(v_p|d) = (1-\lambda_B)\delta_{j,d,P}, p(v_p|d) = (1-\lambda_B)\delta_{j,d,N}$. The only relaxation here is that we allow different topics have their own freedom to choose views, where in CPLSA this view selection distribution is tied for each topic. We further assume that there are M coverages in the collection, each of which corresponds to a document itself. Thus a document can only select the coverage corresponding to itself, κ_d. We can then establish the connection of κ_d with the model parameters in Equation 6.1, such that $p(j|\kappa_d) = \pi_{dj}$. With these transformation and the simple relaxation, it is clear that the topic-sentiment mixture model is an instantiation of the contextual topic model in Chapter 4 (i.e., the CPLSA model).

Please note that since we are not aiming at discovering the content variation of topics in different sentiments, thus we tied the sentiment views so that $\theta_{pj} = \theta_P$, $\theta_{nj} = \theta_N$, and $\theta_{fj} = \theta_j$. It is interesting, however, to relax this constraint and allow the content of a topic to change over different views. We leave this exploration as a future direction.

Similar to existing work [184, 113, 111, 114], we also regularize this model by fixing some parameters. λ_B is set to an empirical constant between 0 and 1, which indicates how much noise that we believe exists

in the weblog collection. We then set the background model as

$$p(w|\theta_B) = \frac{\sum_{d\in\mathcal{D}} c(w,d)}{\sum_{w\in V}\sum_{d\in\mathcal{D}} c(w,d)}$$

The parameters remaining to be estimated are: (1) the topic models, $\Theta = \{\theta_1, ..., \theta_k\}$; (2) the sentiment models, θ_P and θ_N; (3) the document topic probabilities π_{dj}; and (4) the sentiment coverage for each document, $\{\delta_{j,d,F}, \delta_{j,d,P}, \delta_{j,d,N}\}$. We denote the whole set of free parameters as Λ.

Without any prior knowledge, we may use the maximum likelihood estimator to estimate all the parameters. Specifically, we can use the Expectation-Maximization (EM) algorithm [36] to compute the maximum likelihood estimate iteratively; the updating formulas are shown in Figure 6.3. In these formulas, $\{z_{d,w,j,s}\}$ is a set of hidden variables ($s \in \{F, P, N\}$), and $p(z_{d,w,j,s})$ is the probability that word w in document d is generated from the j-th topic, using topic/sentiment model s.

$$
\begin{aligned}
p(z_{d,w,j,F}=1) &= \frac{(1-\lambda_B)\pi_{dj}^{(n)}\delta_{j,d,F}^{(n)}p^{(n)}(w|\theta_j)}{\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j'=1}^{k}\pi_{dj}^{(n)}(\delta_{j',d,F}^{(n)}p^{(n)}(w|\theta'_j) + \delta_{j',d,P}^{(n)}p^{(n)}(w|\theta_P) + \delta_{j',d,N}^{(n)}p^{(n)}(w|\theta_N))}\\
p(z_{d,w,j,P}=1) &= \frac{(1-\lambda_B)\pi_{dj}^{(n)}\delta_{j,d,P}^{(n)}p^{(n)}(w|\theta_P)}{\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j'=1}^{k}\pi_{dj}^{(n)}(\delta_{j',d,F}^{(n)}p^{(n)}(w|\theta'_j) + \delta_{j',d,P}^{(n)}p^{(n)}(w|\theta_P) + \delta_{j',d,N}^{(n)}p^{(n)}(w|\theta_N))}\\
p(z_{d,w,j,N}=1) &= \frac{(1-\lambda_B)\pi_{dj}^{(n)}\delta_{j,d,N}^{(n)}p^{(n)}(w|\theta_N)}{\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j'=1}^{k}\pi_{dj}^{(n)}(\delta_{j',d,F}^{(n)}p^{(n)}(w|\theta'_j) + \delta_{j',d,P}^{(n)}p^{(n)}(w|\theta_P) + \delta_{j',d,N}^{(n)}p^{(n)}(w|\theta_N))}\\
\pi_{dj}^{(n+1)} &= \frac{\sum_{w\in V} c(w,d)(p(z_{d,w,j,F}=1)+p(z_{d,w,j,P}=1)+p(z_{d,w,j,N}=1))}{\sum_{j'=1}^{k}\sum_{w\in V} c(w,d)(p(z_{d,w,j',F}=1)+p(z_{d,w,j',P}=1)+p(z_{d,w,j',N}=1))}\\
\delta_{j,d,F}^{(n+1)} &= \frac{\sum_{w\in V} c(w,d)p(z_{d,w,j,F}=1)}{\sum_{w\in V} c(w,d)(p(z_{d,w,j,F}=1)+p(z_{d,w,j,P}=1)+p(z_{d,w,j,N}=1))}\\
\delta_{j,d,P}^{(n+1)} &= \frac{\sum_{w\in V} c(w,d)p(z_{d,w,j,P}=1)}{\sum_{w\in V} c(w,d)(p(z_{d,w,j,F}=1)+p(z_{d,w,j,P}=1)+p(z_{d,w,j,N}=1))}\\
\delta_{j,d,N}^{(n+1)} &= \frac{\sum_{w\in V} c(w,d)p(z_{d,w,j,N}=1)}{\sum_{w\in V} c(w,d)(p(z_{d,w,j,F}=1)+p(z_{d,w,j,P}=1)+p(z_{d,w,j,N}=1))}\\
p^{(n+1)}(w|\theta_j) &= \frac{\sum_{d\in\mathcal{D}} c(w,d)p(z_{d,w,j,F}=1)}{\sum_{w'\in V}\sum_{d\in\mathcal{D}} c(w',d)p(z_{d,w',j,F}=1)}\\
p^{(n+1)}(w|\theta_P) &= \frac{\sum_{d\in\mathcal{D}}\sum_{j=1}^{k} c(w,d)p(z_{d,w,j,P}=1)}{\sum_{w'\in V}\sum_{d\in\mathcal{D}}\sum_{j=1}^{k} c(w',d)p(z_{d,w',j,P}=1)}\\
p^{(n+1)}(w|\theta_N) &= \frac{\sum_{d\in\mathcal{D}}\sum_{j=1}^{k} c(w,d)p(z_{d,w,j,N}=1)}{\sum_{w'\in V}\sum_{d\in\mathcal{D}}\sum_{j=1}^{k} c(w',d)p(z_{d,w',j,N}=1)}
\end{aligned}
$$

Figure 6.3: EM updating formulas for the topic-sentiment mixture model

However, in reality, if we do not provide any constraint on the model, the sentiment models estimated from the EM algorithm will be very biased towards specific contents of the collection, and the topic models will also be "contaminated" with sentiments. This is because the opinion words and topical words may co-occur with each other, thus they will not be separated by the EM algorithm. This is unsatisfactory as we want our sentiment models to be independent of the topics, while the topic models should be neutral.

In order to solve this problem, we introduce a regularized two-phase estimation framework, in which we first learn a general prior distribution on the sentiment models and then combine this prior with the data likelihood to estimate the parameters using the maximum a posterior (MAP) estimator.

6.3.3 Defining Model Priors

The prior distribution should tell the TSM what the sentiment models should look like in the working collection. This knowledge may be obtained from domain specific lexicons, or training data in this domain as in [134]. However, it is impossible to have such knowledge or training data for every ad hoc topics, or queries. Therefore, we want the prior sentiment models to be general enough to apply to any ad hoc topics. In this section, we show how we may exploit an online sentiment retrieval service such as Opinmind [130] to induce a general prior on the sentiment models. When given a query, Opinmind can retrieve positive sentences and negative sentences, thus we can obtain examples with sentiment labels for a topic (i.e., the query) from Opinmind. The query can be regarded as a topic label. To ensure diversity of topics, we can submit various queries to Opinmind and mix all the results to form a training collection. Presumably, if the topics in this training collection are diversified enough, the sentiment models learnt would be very general.

With such a training collection, we have topic labels and sentiment labels for each document. Formally, we have $\mathcal{D} = \{(d, t_d, s_d)\}$, where t_d indicates which topics the document is about, and s_d indicates whether d holds positive or negative opinions about the topics. We then use the topic-sentiment model presented in Section 6.3.2 to fit the training data and estimate the sentiment models. Since we have topic and sentiment labels, we impose the following constraints: (1) $\pi_{dj} = 1$ if $t_d = j$ and $\pi_{dj} = 0$ otherwise; (2) $\delta_{j,d,P} = 0$ if s_d is *negative* and $\delta_{j,d,N} = 0$ if s_d is *positive*.

In Section 6.5, we will show that this estimation method is effective for extracting general sentiment models and the diversity of topics helps improve the generality of the sentiment models learnt.

Rather than directly using the learnt sentiment models to analyze our target collection, we use them to define a prior on the sentiment models and estimate sentiment models (and the topic models) using the maximum a posterior estimator. This way would allow us to adapt the general sentiment models to our collection and further improve the accuracy of the sentiment models, which is traditionally done in a domain dependent way. Specifically, let $\bar{\theta}_P$ and $\bar{\theta}_N$ be the positive and negative sentiment models learnt from some training collections. We define the following two conjugate Dirichlet priors for the sentiment model θ_P and θ_N, respectively: $Dir(\{1 + \mu_P p(w|\bar{\theta}_P)\}_{w \in V})$ and $Dir(\{1 + \mu_N p(w|\bar{\theta}_N)\}_{w \in V})$, where the parameters μ_P and μ_N indicate how strong our confidence is on the sentiment model prior. Since the prior is conjugate, μ_P (or μ_N) can be interpreted as "equivalent sample size", which means that the impact of adding the prior

would be equivalent to adding $\mu_P p(w|\bar{\theta}_P)$ (or $\mu_N p(w|\bar{\theta}_N)$) pseudo counts for word w when estimating the sentiment model $p(w|\theta_P)$ (or $p(w|\theta_N)$).

If we have some prior knowledge on the topic models, we can also define them as conjugate prior for some θ_j. Indeed, given a topic, a user often has some knowledge about what aspects are interesting. For example, when the user is searching for laptops, we know that he is very likely interested in "price" and "configuration". It will be nice if we "guide" the model to enforce two of the topic models to be as close as possible to the predefined facets.

Therefore, in general, we may assume that the prior on all the parameters in the model is

$$\begin{aligned} p(\Lambda) \quad \propto \quad & p(\theta_P) * p(\theta_N) * \prod_{j=1}^{k} p(\theta_j) = \prod_{w \in V} p(w|\theta_P)^{\mu_P p(w|\bar{\theta}_P)} \\ & \prod_{w \in V} p(w|\theta_N)^{\mu_N p(w|\bar{\theta}_N)} \prod_{j=1}^{k} \prod_{w \in V} p(w|\theta_j)^{\mu_j p(w|\bar{\theta}_j)} \end{aligned}$$

where $\mu_j = 0$ if we do not have prior knowledge on θ_j.

6.3.4 Maximum A Posterior Estimation

With the prior defined above, we may use the MAP estimator: $\hat{\Lambda} = \arg\max_{\Lambda} p(\mathcal{D}|\Lambda)p(\Lambda)$

It can be computed by rewriting the M-step in the EM algorithm in Section 6.3.2 to incorporate the pseudo counts given by the prior [106]. The new M-step updating formulas are:

$$p^{(n+1)}(w|\theta_P) = \frac{\mu_P p(w|\bar{\theta}_P) + \sum_{d \in \mathcal{D}} \sum_{j=1}^{k} c(w,d) p(z_{d,w,j,P} = 1)}{\mu_P + \sum_{w' \in V} \sum_{d \in \mathcal{D}} \sum_{j=1}^{k} c(w',d) p(z_{d,w',j,P} = 1)}$$

$$p^{(n+1)}(w|\theta_N) = \frac{\mu_N p(w|\bar{\theta}_N) + \sum_{d \in \mathcal{D}} \sum_{j=1}^{k} c(w,d) p(z_{d,w,j,N} = 1)}{\mu_N + \sum_{w' \in V} \sum_{d \in \mathcal{D}} \sum_{j=1}^{k} c(w',d) p(z_{d,w',j,N} = 1)}$$

$$p^{(n+1)}(w|\theta_j) = \frac{\mu_j p(w|\bar{\theta}_j) + \sum_{d \in \mathcal{D}} c(w,d) p(z_{d,w,j,F} = 1)}{\mu_j + \sum_{w' \in V} \sum_{d \in \mathcal{D}} c(w',d) p(z_{d,w',j,F} = 1)}$$

The parameters $\mu' s$ can be either empirically set to constants, or set through regularized estimation [165], in which we would start with very large $\mu' s$ and then gradually discount $\mu' s$ in each EM iteration until some stopping condition is satisfied.

6.3.5 Utilizing the Model

Once the parameters in the model are estimated, many tasks can be done by utilizing the model parameters.

1. Rank sentences for topics: Given a set of sentences and a theme j, we can rank the sentences according

to a topic j with the score

$$Score_j(s) = -D(\theta_j||\theta_s) = -\sum_{w \in V} p(w|\theta_j) \log \frac{p(w|\theta_j)}{p(w|\theta_s)}$$

where θ_s is a smoothed language model of sentence s.

2. Categorize sentences by sentiments: Given a sentence s assigned to topic j, we can assign s to positive, negative, or neutral sentiment according to

$$\arg\max_x -D(\theta_s||\theta_x) = \arg\max_x -\sum_{w \in V} p(w|\theta_s) \log \frac{p(w|\theta_s)}{p(w|\theta_x)}$$

where $x \in \{j, P, N\}$, and θ_s is a language model of s.

3. Reveal the overall opinions for documents/topics: Given a document d and a topic j, the overall sentiment distribution for j in d is the sentiment coverage $\{\delta_{j,d,F}, \delta_{j,d,P}, \delta_{j,d,N}\}$. The overall sentiment strength (e.g., positive sentiment) for the topic j is

$$S(j, P) = \frac{\sum_{d \in \mathcal{D}} \pi_{dj} \delta_{j,d,P}}{\sum_{d \in \mathcal{D}} \pi_{dj}}$$

6.4 Sentiment Dynamics Analysis

While the TSM model can be directly used to analyze topics and sentiments in many ways, it does not directly model the topic life cycles or sentiment dynamics. In addition to associating the sentiments with multiple subtopics, we would also like to show how the positive/negative opinions about a given subtopic change over time. The comparison of such temporal patterns (i.e., topic life cycles and corresponding sentiment dynamics) could potentially provide more in-depth understanding of the public opinions than [130], and yield more accurate predictions of user behavior than using the methods proposed in [52] and [123].

To achieve this goal, we can approximate these temporal patterns by partitioning documents into their corresponding time periods and computing the posterior probability of $p(t|\theta_j)$, $p(t|\theta_j, \theta_P)$ and $p(t|\theta_j, \theta_N)$, where t is a time period. This approach has the limitation that these posterior distributions are not well defined, because the time variable t is nowhere involved in the original model. An alternative approach would be to model the time variable t explicitly in the model as in [111, 114], but this would bring in many more free parameters to the model, making it harder to estimate all the parameters reliably. Defining a good partition of the time line is also a challenging problem, since too coarse a partition would miss many

bursting patterns, while too fine granularity a time period may not be estimated reliably because of data sparseness.

In this work, we present another approach to extract topic life cycles and sentiment dynamics, which is similar to the method used in [113]. Specifically, we use a hidden Markov model (HMM) to tag every word in the collection with a topic and sentiment polarity. Once all words are tagged, the topic life cycles and sentiment dynamics could be extracted by counting the words with corresponding labels.

We first sort the documents with their time stamps, and convert the whole collection into a long sequence of words. On the surface, it appears that we could follow [113] and construct an HMM with each state corresponding to a topic model (including the background model), and set the output probability of state j to $p(w|\theta_j)$. A topic state can either stay on itself or transit to some other topic states through the background state. The system can learn (from our collection) the transition probabilities with the Baum-Welch algorithm [141] and decode the collection sequence with the Viterbi algorithm [141]. We can easily model sentiments by adding two sentiment states to the HMM. Unfortunately, this structure cannot decode which sentiment word is about which topic. Below, we present an alternative HMM structure (shown in Figure 6.4) that can better serve our purpose.

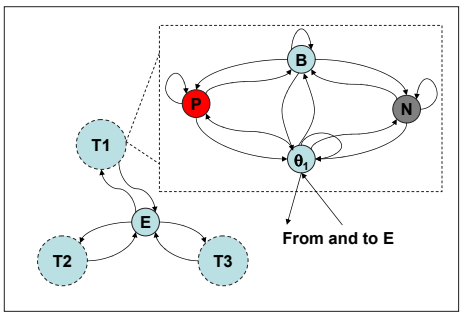


Figure 6.4: The Hidden Markov Model to extract topic life cycles and sentiment dynamics

In Figure 6.4, state E controls the transitions between topics. In addition to E, there are a series of pseudo states, each of which corresponds to a subtopic. These pseudo states can only transit from each other through state E. A pseudo state is not a real single state, but a substructure of states and transitions. For example, pseudo state T1 consists of four real states, three of them correspond to the topic model θ_1, the positive sentiment model, and the negative sentiment model. The remaining state corresponds to the background model. The four states in each pseudo state are fully connected, except that there is no direct transition between two sentiment states. The output probabilities of all states (except for state E) are fixed according to the corresponding topic or sentiment models. This HMM structure can decode both topic segments (with pseudo state T_j) and sentiment segments associated with each topic (with states inside T_j).

We force the model to start with state E, and use the Baum-Welch algorithm to learn the transition probabilities and the output probability for E. Once all the parameters are estimated, we use the Viterbi algorithm to decode the collection sequence. Finally, as in [113], we compute the topic life cycles and sentiment dynamics by counting the number of words labeled with the corresponding state over time.

6.5 Experiments and Results

6.5.1 Data Sets

We need two types of data sets for evaluation. One is used to learn the general sentiment priors, thus should have labels for positive and negative sentiments. In order to extract very general sentiment models, we want the topics in this data set to be as diversified as possible. We construct this training data set by leveraging an existing weblog sentiment retrieval system (i.e., Opinmind.com [130]), i.e., we submit different queries to Opinmind and mix the downloaded classified results. This also gives us natural boundaries of topics in the training collection. The composition of this training data set (denotated as "OPIN") is shown in Table 6.1.

Topic	# Pos.	# Neg.	Topic	# Pos.	# Neg.
laptops	346	142	people	441	475
movies	396	398	banks	292	229
universities	464	414	insurances	354	297
airlines	283	400	nba teams	262	191
cities	500	500	cars	399	334

Table 6.1: Basic statistics of the OPIN data sets

The other type of data is used to evaluate the extraction of topic models, topic life cycles, and sentiment dynamics. Such data do not need to have sentiment labels, but should have time stamps, and be able to represent users' ad hoc information needs. Following [113], we construct these data sets by submitting time-bounded queries to Google Blog Search [1] and collect the blog entries returned. We restrict the search domain to spaces.live.com, since schema matching is not our focus. The basic information of these test collections (notated as "TEST") is shown in Table 6.2.

Data Set	# doc.	Time Period	Query Term
iPod	2988	1/11/05∼11/01/06	ipod
Da Vinci Code	1000	1/26/05∼10/31/06	da+vinci+code

Table 6.2: Basic statistics of the TEST data sets

For all the weblog collections, Krovetz stemmer [75] is used to stem the text.

[1] http://blogsearch.google.com

6.5.2 Sentiment Model Extraction

Our first experiment is to evaluate the effectiveness of learning the prior models for sentiments. As discussed in Section 6.3.3, a good $\bar{\theta}_s$ should not be dependent with the specific features of topics, and be general enough to be used to guild the learning of sentiment models for unseen topics. The more diversified the topics of the training set are, the more general the sentiment models estimated should be.

To evaluate the effectiveness of our TSM model on this task, we collect labeled results of 10 different topics from Opinmind, each of which consists of an average of 5 queries. We then construct a series of training data sets, such that for any k ($1 \leq k \leq 9$), there are 10 training data sets, each of which is a mixture of k topics. We then apply the TSM model on each data set and extract sentiment models accordingly. We also construct a dataset with the mixture of all 10 topics. The top words of the sentiment models which are extracted from the 10-topic-mixture data set and those from a single-topic data set are compared in Table 6.3.

P-mix	**N-mix**	P-movies	N-movies	P-cities	N-cities
love	suck	love	hate	beautiful	hate
awesome	hate	harry	harry	love	suck
good	stupid	pot	pot	awesome	people
miss	ass	brokeback	mountain	amaze	traffic
amaze	fuck	mountain	brokeback	live	drive
pretty	horrible	awesome	suck	good	fuck
job	shitty	book	evil	night	stink
god	crappy	beautiful	movie	nice	move
yeah	terrible	good	gay	time	weather
bless	people	watch	bore	air	city
excellent	evil	series	fear	greatest	transport

Table 6.3: Sentiment models learnt from a mixture of topics are more general

The left two columns in Table 6.3 present the two sentiment models extracted from the 10-topic-mixture dataset, which is more general than the right two columns and two in the middle, which are extracted from two single-topic data sets ("movies" and "cities") respectively. In the two columns in the middles, we see terms like "harry", "pot", "brokeback", "mountain" ranked highly in the sentiment models. These words are actually part of our query terms. We also see other domain specific terms such as "movie", "series", "gay", and "watch". In the sentiment models from "cities" dataset, we remove all query terms from the top words. However, we could still notice words like "night", "air" in the positive model, and "traffic", "weather", "transport" in the negative model. This indicates that the sentiment models are highly biased towards the specific features of the topic, if the training data set only contains one topic.

To evaluate this in a more principled way, we conduct a 10-fold cross validation, which numerically measures the closeness of the sentiment models learnt from a mixture of topics ($k = 1 \sim 9$), and those from an unseen topic (i.e., a topic not in the mixture). Intuitively, a sentiment model is less biased if it is closer to unseen topics. The closeness of two sentiment models is measured with the Kullback-Leibler(KL)

Divergence, the formula of which is

$$D(\theta_x||\theta_y) = \sum_{w \in V} p(w|\theta_x) \log \frac{p(w|\theta_x)}{p(w|\theta_y)}$$

where θ_x and θ_y are two sentiment models (e.g., θ_P learnt from a mixture-topic collection and θ_P from a single-topic collection). We use a simple Laplace smoothing method to guarantee $p(w|\theta_y) > 0$. The result of the cross validation is presented in Figure 6.5.

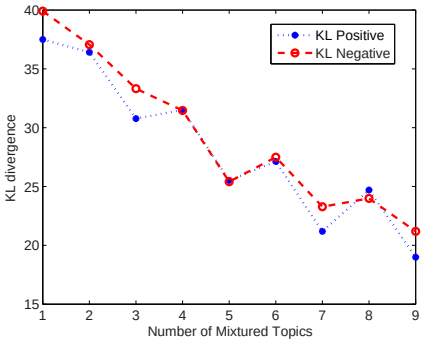


Figure 6.5: Sentiment model leant from diversified topics better fits unseen topics

Figure 6.5 measures the average KL divergence between the positive (negative) sentiment model learnt from a k-topic-mixture dataset and the positive (negative) sentiment model learnt from an unseen single-topic dataset. We notice that when k is larger, where the topics in the training dataset are more diversified, the sentiment models learnt from the collection are closer to the sentiment models of unseen topics. This validates our assumption that a more diversified training collection could provide more general sentiment prior models for new topics. The sentiment models ($\bar{\theta}_P$ and $\bar{\theta}_N$) estimated from the 10-topic-mixture collection are used as the prior sentiment models in the following experiments.

6.5.3 Topic Model Extraction

Our second experiment is to fit the TSM to ad hoc weblog collections and extract the topic models and sentiment coverage. As discussed in Section 6.3.3, the general sentiment models learnt from the OPIN Data set will be used as a strong prior for the sentiment models in a given collection. We expect the topic models extracted be unbiased towards sentiment polarities, which simply represent the neutral contents of the topics.

In the experiments, we set the initial values of $\mu's$ reasonably large (>10,000), and use the regularized estimation strategy in [165] to gradually decay the $\mu's$. λ_B is empirically set between 0.85 $\sim$ 0.95. Some informative topic models extracted from the TEST data sets are shown in Table 6.4, 6.5.

NO-Prior			With-Prior	
batt., nano	**marketing**	**ads, spam**	**Nano**	**Battery**
battery	apple	free	nano	battery
shuffle	microsoft	sign	color	shuffle
charge	market	offer	thin	charge
nano	zune	freepay	hold	usb
dock	device	complete	model	hour
itune	company	virus	4gb	mini
usb	consumer	freeipod	dock	life
hour	sale	trial	inch	rechargeable

Table 6.4: Example topic models with TSM: iPod

NO-Prior			With-Prior	
content	**book**	**background**	**movie**	**religion**
langdon	author	jesus	movie	religion
secret	idea	mary	hank	belief
murder	holy	gospel	tom	cardinal
louvre	court	magdalene	film	fashion
thrill	brown	testament	watch	conflict
clue	blood	gnostic	howard	metaphor
neveu	copyright	constantine	ron	complaint
curator	publish	bible	actor	communism

Table 6.5: Example topic models: Da Vinci Code

As discussed in Section 6.3.4, we can either extract topic models in a completely unsupervised way, or base on some prior of what the topic models should look like. In Table 6.4, 6.5, the left three columns are topic models extracted without prior knowledge, and the right columns are those extracted with the bold titles as priors. We see that the topics extracted in either way are informative and coherent. The ones extracted with priors are extremely clear and distinctive, such as "Nano" and "battery" for the query "iPod". This is quite desirable in summarizing search results, where the system could extract topics in an interactive way with the user. For example, the user can input several words as expected facets, and the system uses these words (e.g., "movie", "book", "history" for the query "Da Vinci Code" and "battery", "price" for the query "iPod") as prior on some topic models, and let the remaining to be extracted in the unsupervised way.

With the topic models and sentiment models extracted, we can summarize the sentences in blog search results, by first ranking sentences according to different topics, and then assigning them into sentiment categories. Table 6.6 shows the summarized results for the query "Da Vinci Code".

We show two facets of the results: "movie" and "book". Although both the movie and the book are named as "The Da Vinci Code", many people hold different opinions about them. Table 6.6 well organizes sentences retrieved for the query "da vinci code" by the relevance to each facets, and the categorization as positive, negative, and neutral opinions. The sentences do not have to contain the facet name, such as "Tom Hanks stars in the movie". The bolded sentence clearly presents an example of mixed topics. We also notice that the system sometimes make the wrong classifications. For example, the sentence "anybody is interested in it?" is misclassified as positive. This is because we rely on a unigram language model for the sentiments,

	Neutral	Thumbs Up	Thumbs Down
Topic1 **Movie**	... Ron Howards selection of Tom Hanks to play Robert Langdon.	Tom Hanks stars in the movie, who can be mad at that?	But the movie might get delayed and even killed off if he loses.
	Directed by: Ron Howard Writing credits: Akiva Goldsman ...	Tom Hanks, who is my favorite movie star act the leading role.	protesting ... will lose your faith by ... watching the movie.
	After watching the movie I went online and some research on ...	Anybody is interested in it?	**... so sick of people making such a big deal about a FICTION book and movie.**
Topic2 **Book**	I knew this because I was once a follower of feminism.	And I'm hoping for a good book too.	**... so sick of people making such a big deal about a FICTION book and movie.**
	I remembered when i first read the book, I finished the book in two days.	Awesome book.	This controversy book cause lots conflict in west society.
	I'm reading "Da Vinci Code" now.	So still a good book to past time.	in the feeling of deeply anxious and fear, to ... read books calmly was quite difficult.

Table 6.6: Topic-sentiment summarization: Da Vinci Code

TSM		Opinmind	
Thumbs Up	**Thumbs Down**	**Thumbs Up**	**Thumbs Down**
(sweat) iPod Nano ok so ...	WAT IS THIS SHIT??!!	I love my iPod, I love my G5...	I hate ipod.
Ipod Nano is a cool design, ...	ipod nanos are TOO small!!	I love my little black 60GB iPod	Stupid ipod out of batteries...
the battery is one serious	Poor battery life ...	I LOVE MY iPOD	" hate ipod " = 489..
example of excellent relibability	... battery completely died	I love my iPod.	iPod looked uglier...surface...
My new VIDEO ipod arrived!!!	fake video ipod	- I love my iPod.	i hate my ipod.
Oh yeah! New iPod video	Watch video podcasts ...	... iPod video looks SO awesome	... microsoft ... the iPod sucks

Table 6.7: Topic-sentiment summarization: iPod

and the "bag of words" assumption does not consider word dependency and linguistics. This problem can be tackled when phrases are used as the bases of the sentiment models.

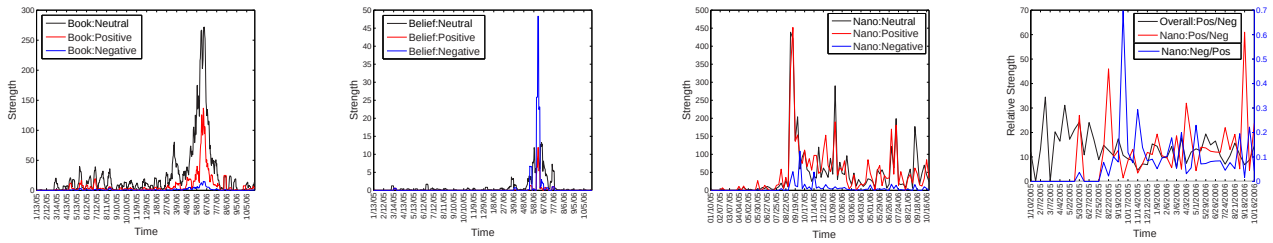
In Table 6.7, we compare the query summarization of our model to that of Opinmind. The left two columns are summarized search results with TSM and right two columns are top results from Opinmind, with the same query "iPod". We see that Opinmind tends to rank sentences with strongest sentiments to the top, but many of which are not very informative. For example, although the sentences "I love iPod" and "I hate iPod" do reflect strong attitudes, they do not give the user as useful information as "out of battery again". Our system, on the other hand, reveals the hidden facets of people's opinions. In the results from Opinmind, we do notice that some sentences are about specific aspects of iPod, such as "battery", "video", "microsoft (indicating marketing)". Unfortunately, these useful information are mixed together. Our system organizes the sentences according to the hidden aspects, which provides the user a deeper understanding of the opinions about the query.

6.5.4 Topic Life Cycle and Sentiment Dynamics

Based on the topic models and sentiment models learnt from the TEST collections, we evaluate the effectiveness of the HMM based method presented in Section 6.4, on extraction of topic life cycles and sentiment dynamics.

Intuitively, we expect the sentiment models to explain as much information as possible, since the most useful patterns are sentiment dynamics. In our experiments, we force the transition probability from topic states to sentiment states, and those from sentiment models to themselves to be reasonably large (e.g.,

>0.25). The results of topic life cycles and sentiment dynamics are selectively presented in Figure 6.6.



(a) Da Vinci Code: Book (b) Da Vinci Code: Religion (c) iPod: Nano (d) iPod: Relative

Figure 6.6: Topic life cycles and sentiment dynamics

In Figure 6.6(a) and (b), we present the dynamics of two facets in the Da-Vinci-Code Collection: "book" and "religion, belief". The neutral line in each plot corresponds to the topic life cycle. In both plots, we see a significant burst in May, 2006, which was caused by the release of the movie "Da Vinci Code". However, before the movie, we can still notice some bursts of discussions about the book. In the plot of the "book" facet, the positive sentiments consistently dominates the opinions during the burst. For the religion issues and the conflicts of the belief, however, the negative opinions are stronger than the positive opinions during the burst, which is consistent with the heated debates about the movie around that period of time. In fact, the book and movie are boycotted or even banned in some countries because of the confliction to their religious belief.

In Figure 6.6(c) and (d), we present the topic life cycle and the sentiment dynamics of the subtopic "Nano" for "iPod". In Figure 6.6(c), we see that both the neutral topic and the positive sentiment about Nano burst around early September, 2005. That is consistent with the time of the official introduction of iPod Nano. The negative sentiment, however, does not burst until several weeks later. This is reasonable, since people need to experience the product for a while before discovering its defects. In Figure 6.6(d), we alternatively plot the relative strength of positive (negative) sentiment over the negative (positive) sentiment. This relative strength clearly reveals which sentiment dominates the opinions, and the trend of this domination. Since there are generally less negative opinions, we plot the Neg/Pos line with a different scale. Again, we see that around the time that the iPod Nano was introduced, the positive sentiments dominate the opinions. However, in October 2005, the negative sentiments shows a sudden increase of coverage. This overlaps with the time period in which there was a bursting of complaints, followed by a lawsuit about the "scratch problem" of iPod Nano.

We also plot the Pos/Neg dynamics of the overall sentiments about "iPod". We see that its shape is much different than the Pos/Neg plot of "Nano". The positive sentiment holds a larger proportion of opinions, but this domination is getting weaker. This also suggests that it is not reasonable to use the overall blog

mentions (not distinguishing subtopics or sentiments), or the general sentiment dynamics (not distinguishing subtopics), to predict the user behavior (e.g., buying a Nano).

All these results show that our method is effective to extract the dynamics of topics and the sentiments.

6.6 Related Work

To the best of our knowledge, modeling the mixture of topics and sentiments has not been addressed in existing work. However, there are several lines of related work.

Weblogs have been attracting increasing attentions from researchers, who consider weblogs as a suitable test bed for many novel research problems and algorithms [76, 53, 52, 111, 123]. Much new research work has found applications to weblog analysis, such as community evolution [76], spatiotemporal text mining [111], opinion tracking [130, 111, 123], information propagation [53], and user behavior prediction [52].

Mei and others introduced a mixture model to extract the subtopics in weblog collections, and track their distribution over time and locations [113]. Gruhl and others [53] proposed a model for information propagation and detect spikes in the diffusing topics in weblogs, and later use the burst of blog mentions to predict spikes of sales of this book in the near future [52]. However, all these models tend to ignore the sentiments in the weblogs, and only capture the general description about topics. This may limit the usefulness of their results. Mishne and others instead used the temporal pattern of sentiments to predict the book sales. Opinmind [130] summarizes the weblog search results with positive and negative categories. On the other hand, researchers also use facets to categorize the latent topics in search results [57]. However, all this work ignores the correlation between topics and sentiments. This limitation is shared with other sentiment analysis work such as [122].

Sentiment classification has been a challenging topic in Natural Language Processing (see e.g., [175, 24]). The most common definition of the problem is a binary classification task of a sentence to either the positive or the negative polarity [134, 132]. Since traditional text categorization methods perform poorly on sentiment classification [134], Pang and Lee proposed a method using mincut algorithm to extract sentiments and subjective summarization for movie reviews [132]. In some recent work, the definition of sentiment classification problem is generalized into a rating scale [133]. The goal of this line of work is to improve the classification accuracy, while we aim at mining useful information (topic/sentiment models, sentiment dynamics) from weblogs. These methods do not either consider the correlation of sentiments and topics or model sentiment dynamics.

Some recent work has been aware of this limitation. Engström studied how the topic dependence in-

fluences the accuracy of sentiment classification and tried to reduce this dependence [40]. In a very recent work [39], the author proposed a topic dependent method for sentiment retrieval, which assumed that a sentence was generated from a probabilistic model consisting of both a topic language model and a sentiment language model. A similar approach could be found in [179]. Their vision of topic-sentiment dependency is similar to ours. However, they do not consider the mixture of topics in the text, while we assume that a document could cover multiple subtopics and different sentiments. Their model requires that a set of topic keywords is given by the user, while our method is more flexible, which could extract the topic models in an unsupervised/semi-supervised way with an EM algorithm. They also requires sentiment training data for every topic, or manually input sentiment keywords, while we can learn general sentiment models applicable to ad hoc topics.

Most opinion extraction work tries to find general opinions on a given topic but did not distinguish sentiments [184, 111]. Liu and others extracted product features and opinion features for a product, thus were able to provide sentiments for different features of a product. However, those product opinion features are highly dependent on the training data sets, thus are not flexible to deal with ad hoc queries and topics. The same problem is shared with [179]. They also did not provide a way to model sentiment dynamics.

There is yet another line of research in text mining, which tries to model the mixture of topics (themes) in documents [61, 10, 113, 111, 114, 90]. The mixture model we presented is along this line. However, none of this work has tried to model the sentiments associated with the topics, thus can not be applied to our problem. However, we do notice that the TSM model is a special case of some very general topic models, such as the CPLSA model [114], which mixes themes with different views (topic, sentiment) and different coverages (sentiment coverages). The generation structure in Figure 6.2 is also related to the general DAG structure presented in [90].

6.7 Summary

In this chapter, we present another instantiation of contextual text mining, with the focus on implicit contexts. Using sentiment as an example, we formally define the problem of topic-sentiment analysis and propose a new probabilistic topic-sentiment mixture model (TSM) to solve this problem. With this model, we could effectively (1) learn general sentiment models; (2) extract topic models orthogonal to sentiments, which can represent the neutral content of a subtopic; and (3) extract topic life cycles and the associated sentiment dynamics. We evaluate our model on different Weblog collections; the results show that the TSM model is effective for topic-sentiment analysis, generating more useful topic-sentiment result summaries for

blog search than a state-of-the-art blog opinion search engine (Opinmind).

The most challenging problem for modeling an implicit context is that the domain of the implicit context is not well defined. In this chapter, we show how to incorporate a prior distribution to help us infer the domain of the implicit context. By adding the prior distribution to the mixture model, we change the maximum likelihood estimation into maximum a posterior (MAP) estimation. We then show how to use the training mode of the TSM model to learn the prior distribution of implicit contexts from the training data.

The exploration in this chapter serves as a guideline one how to model implicit contexts (other than topics) in contextual topic analysis. The component of adding model priors is a general way to incorporate user's prior knowledge as the guidance to the contextual topic model, which elaborates the important component of contextual topic analysis in Section 4.4. Although we use sentiment as a particular example of the implicit context, the proposed approach is general enough to handle other implicit contexts, such as the intents of the user, or the impact of a document.

Chapter 7

Contextual Topic Analysis with Complex Context

In previous chapters, we have introduced how to model explicit contexts as well as implicit contexts like sentiments with topics in contextual topic analysis. In this chapter, we move forward to a more complicated instantiation of contextual text mining - the modeling of complex context.

Contexts are not always independent. There are usually complicated dependency structure among contexts, which makes the contexts themselves a complex system. By the general notion of complex context, we mean the contexts with such a dependency structure, such as a geographic networks, or a social network of people. Without loss of generality, in this chapter we define a complex context as a graph structure of contexts. Unlike the previous chapters, we introduce a new type of regularization over the instantiations of the contextual topic model, by adding a graph-based harmonic regularizer. The regularizer well adapts to the principle that similar contexts have similar generative process.

Like previous chapters, we present this methodology with real world text mining applications. We introduce a problem called topic modeling with network structure, with interesting tasks to discover topical communities and topic maps from the integration of text data and social networks (see [107]).

This chapter gives a detailed discussion about a general component of contextual topic analysis - regularizing contextual topic models with context dependency (Section 4.5), and provides us a general guideline to model explicit contexts in contextual text mining.

7.1 Overview

With the prevailing of Web 2.0 applications, more and more web users are actively publishing text information online. These users also often form social networks in various ways, leading to simultaneous growth of both text information and network structures such as social networks. Taking weblogs (i.e., blogs) as an example, one can find a wide coverage of topics and diversified discussions in the blog posts, as well as a fast evolving friendship network among the bloggers. In another scenario, as researchers are regularly publishing papers, we not only obtain text information, but also naturally have available co-authorship networks of

authors. In yet another scenario, as email users produce many text messages, they also form networks through the relation of sending or replying to messages. One can easily imagine many other examples of text accompanied by network structures such as webpages accompanied by links and literature accompanied by citations. Figure 7.1 presents an example coauthor network from SIGIR proceedings, where each author is associated with the papers he/she published.

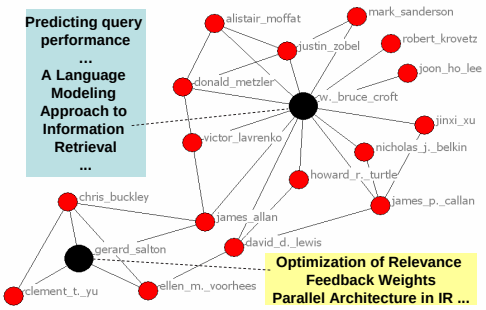


Figure 7.1: A sample network structure with text

These examples show that in many web mining tasks, we are often dealing with collections of text with a network structure attached, making it interesting to study how we can leverage the associated network structure to discover interesting topic and/or network patterns. These network structures can be regarded as complex relational context associated with the text collection. Thus, they can potentially leveraged to enhance text mining if we can incorporate them into a text mining model.

Statistical topic models have recently been successfully applied to multiple text mining tasks [61, 10, 184, 162, 114, 90, 171] to discover a number of topics from text. Some recent work has incorporated into topic modeling context information [114], such as time [171], geographic location [111], and authorship [162, 149, 111], to facilitate contextual text mining. Topics discovered in this way can be used to infer research communities [162, 149] or information diffusion over geographic locations [111]. However, they do not consider the natural network structure among authors, or geographic locations. Intuitively, these network structures are quite useful for refining and structuring topics, and are sometimes essential for discovering network-associated topics. For example, two researchers who often coauthor with each other are likely to be working on the same topics, thus are likely to be in the same research community. For geographically sensitive events (e.g., hurricane Katrina), bloggers living at adjacent locations tend to write about similar topics. The lack of consideration of network structures is also a deficiency in some other text mining techniques such as document clustering.

On the other hand, social network analysis (SNA) focuses on the topology structure of a network [71, 88, 3, 78], addressing questions such as "what the diameter of a network is [88]", "how a network evolves

[88, 3]", "how information diffuses on the network [53, 89]", and "what are the communities on a network [71, 3]." However, these techniques usually do not leverage the rich text information. In many scenarios, text information is very helpful for SNA tasks. For example, Newton and Einstein had never collaborated on a paper (i.e., *social network*), but we still consider them in the same research (physics) community because they made research contributions on related research topics (i.e., *text*). Similarly, to attract Bruce Willis to take a role in a movie, "the script is interesting (i.e., *text*)" is as important as "the director is a trustable friend (i.e., *social network*)."

Is there a way to leverage the power of both the textual topics and the network structure in text mining? Can the two successful complementary mining techniques (i.e., probabilistic topic modeling and social network analysis) be combined to help each other? To the best of our knowledge, these questions have never been seriously studied before. As a result, there is no principled way to combine the mining process of topics in text and social networks (e.g., combining topic modeling with network analysis). Although methods have been proposed to combine page contents and links in web search [146], none of them is tuned for text mining.

In this chapter, we formally define the major tasks of **Topic Modeling with Network Structure (TMN)**, and propose a unified framework to combine statistical topic modeling with network analysis by regularizing the topic model with a discrete regularizer defined based on the network structure. The framework makes it possible to cast our mining problem as an optimization problem with an explicit objective function. Experiment results on different genres of real world data show that our model can effectively extract topics, generat topic maps on a network, and discover topical communities. The results also show that our model improves over both pure text mining methods and pure network analysis methods, suggesting the necessity of combining them.

The proposed framework of regularized topic modeling is general; one can choose any topic model and a corresponding regularizer on the network. Variations of the general model are effective for solving real world text mining problems, such as author-topic analysis and spatial topic analysis.

The rest of this chapter is organized as follows. In Section 7.2, we formally define the problem of topic modeling with network structure. In Section 7.3, we propose the unified regularization framework as well as two general methods to solve the optimization problem. We discuss the variations and applications of our model in Section 7.4 and present empirical experiments in Section 7.5. Finally, we discuss the related work in Section 7.6 and conclude in Section 7.7.

7.2 Problem Formulation

We assume that the data to be analyzed consists of both a collection of text documents and an associated network structure. This setup is quite general: The text documents can be a set of web pages, blog articles, scientific literature, emails, or profiles of web users. The network structure can be any social networks on the web, co-author/citation graphs, geographic networks, or even latent networks that can be inferred from the text (e.g., entity-relation graph, document nearest-neighbor graph, etc). We now formally define the related concepts and the general tasks of topic modeling with network structure.

Definition 7.1 (Network): A ***network*** associated with a text collection $\mathcal{D}$ is a graph $G = \langle V, E \rangle$, where V is a set of vertices and E is a set of edges. Without losing generality, we define a ***vertex*** $u \in V$ as a subset of documents $\mathcal{D}_v \subset \mathcal{D}$. For example, a vertex in a coauthor graph can be a single author associated with all papers he/she published. An ***edge*** $\langle u, v \rangle$ is a binary relation between vertices u and v, where we use $w(u, v)$ to denote the weight of $\langle u, v \rangle$. An edge can be either **undirected** or **directed**. In this work, we only consider the undirected case, i.e., $\langle u, v \rangle = \langle v, u \rangle$.

By combining text topics and a network structure, we can discover new types of interesting patterns. For example, we can explore who first brought the topic "language modeling" into the IR community, and who have been diffusing this topic on the research network. A topic could also define a latent community on the network (e.g., the machine learning community, the SNA community, etc). The following patterns are unique to topic modeling with network structure, and cannot be discovered solely from text or social networks.

Definition 7.2 (Topic Map): A ***topic map*** of a topic θ on network G, M_θ, is represented by a vector of weights $\langle f(\theta, v_1), f(\theta, v_2), ..., f(\theta, v_m) \rangle$, where $v_i \in V$, and $f(\theta, v)$ is a weighting function of a topic on a vertex. For example, we may define f as $f(\theta, v_i) = p(\theta|v_i)$, where $\sum_\theta p(\theta|v_i) = 1$ for all v_i. From a topic map, we can learn how a topic is distributed on the network. Intuitively, we expect that the adjacent vertices be associated with similar topics and the weights of topics on adjacent vertices are similar.

Definition 7.3 (Topical Community): A ***topical community*** on network G is represented by a subset of vertices $\mathcal{V}_\theta \subset V$. We can assign a vertex v to $\mathcal{V}_\theta$ with any reasonable criterion, e.g. $f(\theta, v) > \epsilon$, or $\forall \theta', f(\theta, v) > f(\theta', v)$. The topic model θ is then a natural summary of the semantics of the topical community $\mathcal{V}_\theta$. Intuitively we expect that the vertices within the same topical community are tightly connected and all have a large $f(\theta, v)$; vertices from different topical communities are loosely connected and have different $f(\theta, v)$. A *topical community* is different from a *community* in the SNA literature in that it must have coherent semantics, and can be summarized with a coherent *topic* in text.

Based on the definitions of these concepts, we can formalize the major tasks of **topic modeling with**

network structure (TMN) as follows:

Task 1: (Topic Extraction) Given a collection $\mathcal{D}$ and a network structure G, the task of *Topic Extraction* is to model and extract k major topic models, $\{\theta_1, ..., \theta_k\}$, where k is a user specified parameter.

Task 2: (Topic Map Extraction) Given a collection $\mathcal{D}$ and a network structure G, the task of *Topic Map Extraction* is to model and extract the k weight vectors $\{M_{\theta_1}, ..., M_{\theta_k}\}$, where each vector M_θ is a map of topic θ on network G.

Task 3: (Topical Community Discovery) Given a collection $\mathcal{D}$ and a network structure G, the task of *Topical Community Discovery* is to extract k topical communities $\{\mathcal{V}_1, ..., \mathcal{V}_k\}$, where each $\mathcal{V}_i$ has a coherent semantic summary θ_i, which is one of the k major topics in $\mathcal{D}$.

These tasks are challenging in many ways. First, there is no existing unified model that can embed a network structure in a topic model. Indeed, whether a social network structure can help extracting topics is an open question. Second, in existing community discovery methods, there is no guarantee that the semantics of a community is coherent. It is rather unclear how to satisfy the topical coherency and the connectivity coherency at the same time. Moreover, since it is usually hard to create training examples to this problem, the solution has to be unsupervised.

The three major tasks above are by no means the only tasks of topic modeling with network structure. With the output of such basic tasks, more in-depth analysis can be done. For example, one can compare topic maps over time and analyze how topics are propagating over the network. One can also track the evolution of topical communities.

7.3 Regularizing Topic Models with Network Structure

In this section, we propose a novel and general framework of regularizing statistical topic models with the network structure.

7.3.1 Contextual Topic Models

We first discuss the basic statistical topic models, which have been applied to many text mining tasks [61, 10, 162, 111, 90, 171, 114]. The basic idea of these models is to model documents with a finite mixture model of k topics and estimate the model parameters by fitting the data with the model. Two basic statistical topic models are the Probabilistic Latent Semantic Analysis (PLSA) [61] and the Latent Dirichlet Allocation

(LDA) [10]. For example, the log likelihood of a collection $\mathcal{D}$ to be generated with PLSA is given as follows:

$$L(\mathcal{D}) = \sum_d \sum_w c(w,d) \log \sum_{j=1}^{k} p(\theta_j|d)p(w|\theta_j) \tag{7.1}$$

The parameters in PLSA are $\Psi = \{p(\theta_j|d), p(w|\theta_j)\}_{d,w,j}$. Naturally, we can use $\{p(\theta_j|v)\}_j$ as the weights of topics on vertex v, and compute $p(\theta_j|v)$ by

$$p(\theta_j|v) = \sum_{d \in \mathcal{D}_v} p(\theta_j|d)p(d|v) \tag{7.2}$$

PLSA thus provides an over-simplified solution to the problem of TMN by ignoring the network structure. There is no guarantee that vertices in the same topical community are well connected, or adjacent vertices are associated with similar topics. Indeed, a limitation of PLSA is that there is no constraint on the parameters $\{p(\theta_j|d)\}$ for different d, the number of which grows linearly with the data. Therefore, the parameters $\{p(\theta_j|d)\}_{d,j}$ would overfit the data. To alleviate this overfitting problem, LDA assumes that the document-topic distributions $\{p(\theta_j|d)\}_j$ of each document d are all generated from the same Dirichlet distribution.

Note that PLSA is a special case of the contextual topic model, specifically a special case of the CPLSA model. The contexts are topics and documents themselves. In this section, we use PLSA as an example of contextual topic model and show how we can regularize the model with the dependency structure of the contexts (documents). This regularization framework is however very general and can be applied on any other contextual language models where a dependency structure of the context is involved.

7.3.2 The Regularization Framework

We propose a new framework to model topics with a network structure, by regularizing a statistical topic model with a regularizer on the network. The criterion of this regularization is succinct and natural: vertices which are connected to each other should have similar weights of topics ($f(\theta_j, v)$).

Formally, we define a regularized data likelihood as

$$O(\mathcal{D}, G) = -(1-\lambda)L(\mathcal{D}) + \lambda R(\mathcal{D}, G) \tag{7.3}$$

where $L(\mathcal{D})$ is the log likelihood of the collection $\mathcal{D}$ to be generated by the statistical topic model, and $R(\mathcal{D}, G)$ is a harmonic regularizer defined on the network structure G.

This regularization framework is quite general. We can use any statistical topic model to refine $L(\mathcal{D})$,

and use any graph based regularizer $R(\mathcal{D}, G)$ as long as it can smooth the topics among adjacent vertices. We abbreviate the network regularized statistical topic model as **NetSTM**.

To illustrate this framework, in this chapter we use PLSA as the statistical topic model and a regularizer similar to the graph harmonic function in [189], i.e.,

$$R(\mathcal{D}, G) = \frac{1}{2} \sum_{\langle u,v \rangle \in E} w(u,v) \sum_{j=1}^{k} (f(\theta_j, u) - f(\theta_j, v))^2 \tag{7.4}$$

Correspondingly, we call this model **NetPLSA**.

Note that the regularizer in Equation 7.4 is an extension of the graph harmonic function in [189] to multiple classes (topics). It can be rewritten as

$$R(\mathcal{D}, G) = \frac{1}{2} \sum_{j=1}^{k} f_j^T \Delta f_j \tag{7.5}$$

where f_j is a $|V|$ dimensional vector of the weights of the j-th topic on each vertex (e.g., $\{p(\theta_j|v)\}_v$). Δ is the graph Laplacian matrix [189, 188]. We have $\Delta = D - W$, where W is the matrix of edge weights, and D is a diagonal matrix where $d(u,u) = \sum_v w(u,v)$.

This framework is a general one that can leverage the power of both the topic model and the graph Laplacian regularization. Intuitively, the $L(\mathcal{D})$ in Equation 7.6 measures how likely the data is generated from this topic model. By minimizing $-L(\mathcal{D})$, we will find $\{p(\theta_j|d)\}$ and $\{p(w|\theta_j)\}$ which fits the text data as much as possible. By minimizing $R(\mathcal{D})$, we smooth the topic distributions on the network structure, where adjacent vertices have similar topic distributions.

Although theoretically $f(\theta, u)$ can be defined as any weighting function of a topic θ on u, in practice it must be a function of the parameters in PLSA (i.e., $\{p(\theta_j|d)\}$ and $\{p(w|\theta_j)\}$). When a vertex have multiple documents, an example choice is $f(\theta, u) = p(\theta|u) \propto \sum_{d \in \mathcal{D}_u} p(\theta|d)p(d|u)$.

The parameter λ can then be set between 0 to 1 to control the balance between the data likelihood and the smoothness of topic distributions over the network. It is easy to show that if $\lambda = 0$, the objective function boils down to the log likelihood of PLSA. Minimizing $O(\mathcal{D}, G)$ will give us the topics which best fit the content of the collection. When $\lambda = 1$, this objective function boils down to $\frac{1}{2} \sum_{j=1}^{k} f_j^T \Delta f_j$. Embedded with additional constraints, this is related to the objective of spectral clustering (i.e., ratio cut [17]). By minimizing $O(\mathcal{D}, G)$, we will extract document clusters solely based on the network structure.

An interesting simplified case is when every vertex only contains one document (thus substitute u, v with

d_u, d_v) and $f(\theta, u) = p(\theta|d_u)$. Then we have

$$\begin{aligned} O(\mathcal{D}, G) &= -(1-\lambda) * \sum_d \sum_w c(w,d) \log \sum_{j=1}^{k} p(\theta_j|d) p(w|\theta_j) \\ &+ \frac{\lambda}{2} \sum_{\langle u,v \rangle \in E} w(u,v) \sum_{j=1}^{k} (p(\theta_j|d_u) - p(\theta_j|d_v))^2. \end{aligned} \tag{7.6}$$

In the following section, we discuss parameter estimation of the NetPLSA model in such a simplified case. The estimation for more complex cases can be done similarly.

7.3.3 Parameter Estimation

Let us first consider the special case when $\lambda = 0$. In such a case, the objective function degenerates to the log-likelihood function of PLSA with no regularization.

The standard way of parameter estimation for PLSA is to apply the Expectation Maximization (EM) algorithm [36] which iteratively computes a local maximum of $L(\mathcal{D})$. Specifically, in the E-step, it computes the expectation of the complete likelihood $Q(\Psi; \Psi_n)$, where Ψ denotes all the parameters, and Ψ_n denotes the value of Ψ estimated in the last (n-th) EM iteration. In the M-step, the algorithm finds a better estimate of parameters, Ψ_{n+1}, by maximizing $Q(\Psi; \Psi_n)$:

$$\Psi_{n+1} = \arg\max_{\Psi} Q(\Psi; \Psi_n) \tag{7.7}$$

Computationally, the E-step boils down to computing the conditional distribution of the hidden variables given the data and Ψ_n. The hidden variables in PLSA correspond to the events that a term w in document d is generated from the j-th topic. Formally, we have the **E-Step:**

$$\begin{aligned} z(w,d,j) &= p(\theta_j|w,d,\Psi_n) \\ &= \frac{p_n(\theta_j|d) p_n(w|\theta_j)}{\sum_{j'=1}^{k} p_n(\theta_{j'}|d) p_n(w|\theta_{j'})} \end{aligned} \tag{7.8}$$

$$Q(\Psi; \Psi_n) = \sum_d \sum_w c(w,d) \sum_j z(w,d,j) \log p(\theta_j|d) p(w|\theta_j) \tag{7.9}$$

The maximization problem in the **M-Step** (i.e., Equation 7.7) has a closed form solution:

$$p_{n+1}(\theta_j|d) = \frac{\sum_w c(w,d)z(w,d,j)}{\sum_w \sum_{j'} c(w,d)z(w,d,j')} \tag{7.10}$$

$$p_{n+1}(w|\theta_j) = \frac{\sum_d c(w,d)z(w,d,j)}{\sum_d \sum_w c(w,d)z(w,d,j)}. \tag{7.11}$$

We now discuss how we can extend this standard EM algorithm to handle the case $\lambda \neq 0$. Using a similar derivation to that of the EM algorithm, we have the following expected complete likelihood function for NetPLSA, where for convenience of discussion, we also added the Lagrange multipliers corresponding to the constraints on our parameters:

$$\begin{aligned} Q(\Psi;\Psi_n) &= (1-\lambda)\sum_d \sum_w c(w,d) \sum_j z(w,d,j) \log p(\theta_j|d)p(w|\theta_j) \\ &+ \sum_d \alpha_d(\sum_j p(\theta_j|d) - 1) + \sum_j \alpha_j(\sum_w p(w|\theta_j) - 1) \\ &- \frac{\lambda}{2} \sum_{\langle u,v\rangle \in E} w(u,v) \sum_{j=1}^{k} (p(\theta_j|d_u) - p(\theta_j|d_v))^2 \end{aligned} \tag{7.12}$$

where $\alpha_d(\sum_j p(\theta_j|d) - 1)$ and $\alpha_j(\sum_w p(w|\theta_j) - 1)$ are Lagrange multipliers corresponding to the constraints that

$\sum_j p(\theta_j|d) = 1$ and $\sum_w p(w|\theta_j) = 1$.

Thus in general, we can still use the EM algorithm to estimate the parameters when $\lambda > 0$ in Equation 7.6 by maximizing $-O(\mathcal{D}, G)$. It is easy to see that NetPLSA shares the same hidden variables with PLSA, and the conditional distribution of the hidden variables can still be computed using Equation 7.8. Thus the E-step remains the same.

The M-step is more complicated due to the introduction of the regularizer. The estimation of $P(w|\theta_j)$ does not rely on the regularizer, thus can still be computed using Equation 7.10. Unfortunately, we do not have a closed form solution to re-estimate the parameters $\{P(\theta_j|d)\}_{j,d}$ through maximizing $Q(\Psi;\Psi_n)$.

To solve this problem, we can apply a Newton-Raphson method to update Ψ_{n+1} by finding a local maximum of $Q(\Psi;\Psi_n)$ in the M step. Specifically, let X be the vector of variables to be updated with the Newton-Raphson method (i.e., $\{P(\theta_j|d)\}_{j,d}$ and $\{\alpha_d\}_d$). The updating formula of the Newton-Raphson's method is as follows:

$$X^{(t+1)} = X^{(t)} - \gamma[HQ(X^{(t)};\Psi_n)]^{-1} \bigtriangledown Q(X^{(t)};\Psi_n) \tag{7.13}$$

where $X^{(t)}$ is the new estimation of parameters at the t-th inner iteration of the M step. $\bigtriangledown Q(X;\Psi_n)$ is the

gradient of $Q(X;\Psi_n)$. $HQ(X;\Psi_n)$ is the Hessian matrix of $Q(X;\Psi_n)$. γ is a small step size to ensure the satisfaction of the Wolfe conditions [1]. A good selection of γ guarantees that $p(\theta|d) \geq 0$. It is easy to evaluate all the elements in $HQ(X^{(t)};\Psi_n)$ and $\bigtriangledown Q(X^{(t)};\Psi_n)$. We omit the details due to the limit of space. Instead of computing $[HQ(X^{(t)};\Psi_n)]^{-1}$, it is more efficient to find $X^{(t+1)}$ directly by solving the linear system

$$HQ(X^{(t)};\Psi_n)(X^{(t)} - X^{(t+1)}) = \gamma \bigtriangledown Q(X^{(t)};\Psi_n) \tag{7.14}$$

We want to set the start point of $X^{(0)}$ corresponding to Ψ_n. This is because, to guarantee that the generalized EM algorithm will converge, we need to assure $Q(\Psi_{n+1};\Psi_n) \geq Q(\Psi_n;\Psi_n)$. By setting the start point of Newton-Raphson method at Ψ_n, we ensure that Q would not drop.

7.3.4 An Efficient Algorithm

In the previous section, we give a way to gradually approach the local maximum of $Q(\Psi;\Psi_n)$ at M step. However, this involves multiple iterations of Newton-Raphson updating, in each of which we need to solve a linear system of $|d| * (k+1)$ variables. This significantly increases the cost of parameter estimation of NetPLSA.

In this section, we propose a simpler algorithm for parameter estimation based on the generalized EM algorithm (GEM) [127]. According to GEM, we do not have to find the local maximum of $Q(\Psi_{n+1};\Psi_n)$ at every M step; instead, we only need to find a better value of Ψ in the M-step, i.e., to ensure $Q(\Psi_{n+1};\Psi_n) \geq Q(\Psi_n;\Psi_n)$.

Thus our idea is to optimize the likelihood part and the regularizer part of the objective function separately in hope of finding an improvement of the current Ψ. Specifically, let us write $Q(\Psi;\Psi_n) = (1-\lambda)L'(\mathcal{D}) - \lambda R(\mathcal{D},G)$, where $L'(\mathcal{D})$ denotes the expectation of the complete likelihood of the topic model. Clearly, $Q(\Psi;\Psi_n) \geq Q(\Psi_n;\Psi_n)$ holds if $\Psi = \Psi_n$. We introduce $\Psi_{n+1}^{(0)} = \Psi_n$, which is the first eligible set of parameter values that assure $Q(\Psi;\Psi_n) \geq Q(\Psi_n;\Psi_n)$.

At every M-step, we would first attempt to find $\Psi_{n+1}^{(1)}$ to maximize $L'(\mathcal{D})$ instead of the whole $Q(\Psi;\Psi_n)$. This can be done by simply applying Equation 7.11 and 7.10. Clearly, $Q(\Psi_{n+1}^{(1)};\Psi_n) \geq Q(\Psi_n;\Psi_n)$ does not necessarily hold as the regularizer part may have been decreased. Thus we further start from $\Psi_{n+1}^{(1)}$ and attempt to increase $-R(\mathcal{D},G)$.

The Hessian matrix of $2R(\mathcal{D},G)$ is the graph Laplacian matrix (i.e., $\Delta = D - W$). By applying one Newton-Raphson step on $R(\mathcal{D},G)$, we propose a closed form solution for $\Psi_{n+1}^{(2)}$; we then repeatedly obtain

[1] http://en.wikipedia.org/wiki/Newton%27s_method_in_optimization

$\Psi_{n+1}^{(3)}$, ..., $\Psi_{n+1}^{(m)}$ using the Equation 7.15 until the value of the Q-function starts to drop:

$$p_{n+1}^{(t+1)}(\theta_j|d_u) = (1-\gamma)p_{n+1}^{(t)}(\theta_j|d_u) + \gamma\frac{\sum_{\langle u,v\rangle\in E} w(u,v)p_{n+1}^{(t)}(\theta_j|d_v)}{\sum_{\langle u,v\rangle\in E} w(u,v)} \tag{7.15}$$

Clearly, $\sum_j p_{n+1}^{(t+1)}(\theta_j|d) = 1$ and $p_{n+1}^{(t+1)}(\theta_j|d) \geq 0$ always hold in Equation 7.15. When the step parameter γ is set to 1, it means that the new topic distribution of a document is the average of the old distributions from its neighbors. This is related to the random-walk interpretation in [189]. Every iteration of Equation 7.15 makes the topic distributions smoother on the network. Note that an inner iteration does not affect the estimation of $\{p(w|\theta)\}_{w,\theta}$ in $\Psi_{n+1}^{(1)}$.

The stepping parameter γ can be interpreted as a controlling factor of smoothing the topic distribution among the neighbors. Once we have found $Q(\Psi_{n+1}^{(t)}; \Psi_n) \geq Q(\Psi_n; \Psi_n)$, we can limit the further iterations of processing of Equation 7.15, so that Ψ_{n+1} would not be too far away from $\Psi_{n+1}^{(0)}$.

7.4 Applications of NetSTM

The framework defined in Equation 7.3 is quite general. Actually, one can use any topic models for $L(\mathcal{D})$ and a related regularizer for $R(\mathcal{D}, G)$. The choice of the topic model and the regularizer should be task dependent. In this section, we show that with difference choices of L and R, this framework can be applied to different mining tasks.

7.4.1 Author-Topic Analysis and Community Discovery

Author-topic analysis has been proposed in text mining literature [162, 149, 114]. One major task of author-topic analysis is to extract research topics from scientific literature and to measure the associations between topics and authors. This can be regarded as modeling topic maps and discovering research communities solely based on textual contents, where the authors in the same community works on the same topic. With a topic model, one can find a summary for a topical community, e.g., using the distribution $p(w|\theta)$.

On the other hand, many methods have been proposed to discover communities from social networks [71, 3], which solely explore the network structure. One concrete example is to discover research communities based on the coauthor relationship between researchers, where authors with collaborations are likely to lie in the same community.

However, both directions have their own limitations. In author-topic analysis, the associations between authors are indirectly modeled through the content. A professor and his fellow student may be assigned to two different communities, if they have different flavor of topics. On the other hand, solely relying on the

network structure is at risk of assigning a biologist and a computer scientist into the same community, even if they just coauthored one paper of bioinformatics. Moreover, it is difficult to summarize the semantics of a community (i.e., to explain why they form a community).

To leverage the information in the text and the network, we can apply the NetPLSA model to extract topical communities. Specifically, for each author a, we may concatenate all his/her publications to form a virtual document of a. Then a coauthor social network G is constructed where there is an edge between author a and a' if they coauthored at least one paper.

We can define $w(a, a')$ as the number of papers that a and a' coauthored. Equation 7.6 can be rewritten as

$$\begin{aligned} O(\mathcal{D}, G) &= -(1-\lambda) * \sum_{a} \sum_{w} c(w, a) \log \sum_{j=1}^{k} p(\theta_j|a) p(w|\theta_j) \\ &+ \frac{\lambda}{2} \sum_{\langle a, a' \rangle \in E} w(a, a') \sum_{j=1}^{k} (p(\theta_j|a) - p(\theta_j|a'))^2 \end{aligned} \tag{7.16}$$

Apparently, the model in Equation 7.16 is another special case of the CPLSA model. Here the explicit contexts besides topics are not documents anymore, but different authors. There is still only one unique view of topics, but there are $|\mathcal{A}|$ number of unique coverages, each of which corresponds to an author.

By minimizing $O(\mathcal{D}, G)$, we can estimate $p(\theta_j|a)$, which denotes the probability that the author a belongs to the j-th topical community. The estimated distribution $p(w|\theta_j)$ can be used as a semantic summary of the j-th community.

7.4.2 Spatial Topic Analysis

A general task in spatial text mining [114] is to extract topics from text with location labels and model their distribution over different geographic locations. Some natural topics, like public reaction to an event (e.g., hurricane Katrina), are geographic correlated. Intuitively we can expect that people live at nearby locations express similar topics.

Let L be a set of geographic locations and $l, l' \in L$. We denote $d \in \mathcal{D}_l$ if document d has a location label of l. By introducing a vertex for every location and an edge between two adjacent locations, we construct a geographic location network $G = \langle L, E \rangle$. We can then model the geographic topic distribution with a variation of the NetPLSA, where

$$O(\mathcal{D}, G) = -(1-\lambda) * \sum_d \sum_w c(w,d) \log \sum_{j=1}^{k} p(\theta_j|d) p(w|\theta_j) \quad (7.17)$$
$$+ \frac{\lambda}{2} \sum_{\langle l,l' \rangle \in E} w(l,l') \sum_{j=1}^{k} (\sum_{d \in \mathcal{D}_l} \frac{p(\theta_j|d)}{|l|} - \sum_{d' \in \mathcal{D}_{l'}} \frac{p(\theta_j|d')}{|l'|})^2$$

where $|l|$ is the number of documents in l. Specifically, we modify the regularizer by replacing $p(\theta_j|d)$ with $\sum_{d \in \mathcal{D}_l} \frac{p(\theta_j|d)}{|l|}$, which is the topic distribution over a location, instead of a single document (we assume a uniform $p(d|l)$). It is not hard to show that the contextual topic model is another special case of CPLSA.

We then use the following formula instead of Equation 7.15:

$$p_{n+1}^{(t+1)}(\theta_j|d) = (1-\gamma) p_{n+1}^{(t)}(\theta_j|d) \quad (7.18)$$
$$+ \gamma \frac{\sum_{\langle l_d, l' \rangle \in E} w(l_d, l') \sum_{d' \in \mathcal{D}_{l'}} \frac{p_{n+1}^{(t)}(\theta_j|d')}{|l'|}}{\sum_{\langle d, d' \rangle \in E} w(l_d, l')}.$$

where l_d denotes the location which d belongs to.

7.5 Experiments

In the previous sections, we introduced the novel framework of topic modeling with network regularization, and discussed how it could be applied to solve real world text mining problems. In this section, we show the effectiveness of our model with experiments on two genres of data. We show how NetPLSA works for the author-topic analysis in Section 7.5.1 and for spatial topic analysis in Section 7.5.2.

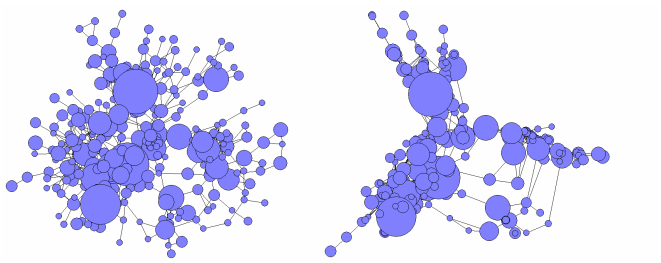
7.5.1 DBLP Author-Topic Analysis

Data Collection

The Digital Bibliography and Library Project (DBLP) is a database which contains the basic bibliographic information of computer science publications [2]. In this experiment, we create our testing data set (4-CONF) from a subset of the DBLP records. We first extract all the papers published at four different conferences, WWW, SIGIR, KDD, and NIPS. For each paper, we extract the title in text and all its authors. We then construct the coauthor network, by making a vertex for every unique author a and an edge $\langle a, a' \rangle$ between two authors if they have coauthored at least one paper. We weight each edge in this network, by the number of papers that the two researchers have coauthored, $w(a, a')$. Finally, we concatenate the titles of all papers

[2] http://www.informatik.uni-trier.de/~ley/db/

of an author to create a document d_a associated with this author. Our dataset has 9041 authors, and 16902 unique edges (without self links); the average weight for an edge is 1.2.



(a) View I (b) View II

* In this figure, we only show the authors who have more than 7 publications in the four conference(s). We do not show singletons.

Figure 7.2: Coauthor network in DBLP dataset

In Figure 7.2, we visualize the coauthor network structure of the 4-CONF dataset using the NetDraw software[3]. We only show the authors with more than five publications. The two views are "*Spring Embedder*" and "*Gower Metric Scaling*" provided by NetDarw. Basically, *Spring Embedder* is a standard graph layout algorithm which tries to put two vertices which are connected by an edge closer, and *Gower Metric Scaling* will locate two vertices closer if they are intensely connected directly or through other vertices [12]. Therefore, in both layout views (Figure 7.2 (a) and (b)), authors closer to each other are more likely to be in the same community. Clearly, from Figure 7.2 (b), we can guess that there are 3 to 4 major communities in the 4-CONF dataset, and such major communities are connected.

Topic Extraction

Once we created the testing datasets, we extract topics from the data using both PLSA and NetPLSA. Since the testing data is a mixture of four conferences, it is interesting to see whether the extracted topics could automatically reveal this mixture. Therefore, in both PLSA and NetPLSA, we set the number of topics to be 4. Following [184, 111], we introduce an extra background topic model to absorb the common words in English. We run the EM algorithm multiple times with random starting points to improve the local maximum of the EM estimates. To make the comparison fair, we use the same starting points for PLSA and NetPLSA. We summarize each topic θ with terms having the highest $p(w|\theta)$.

From Table 7.1, we see that PLSA extracts reasonable topics. However, in terms of representing research communities, all four topics have their limitations. The first topic is somewhat related to information retrieval, but it is mixed with some heterogenous topic like "*protein*". Although the third column is a very coherent NIPS topic (i.e., analog VLSI of neural networks), it is not broad enough to represent the general

[3]http://www.analytictech.com/Netdraw/netdraw.htm

community of NIPS.

Table 7.1: Topics extracted from 4-CONF with PLSA

Topic 1		Topic 2		Topic 3		Topic 4	
term	0.02	peer	0.02	visual	0.02	interface	0.02
question	0.02	patterns	0.01	analog	0.02	towards	0.02
protein	0.01	mining	0.01	neurons	0.02	browsing	0.02
training	0.01	clusters	0.01	vlsi	0.01	xml	0.01
weighting	0.01	streams	0.01	motion	0.01	generation	0.01
multiple	0.01	frequent	0.01	chip	0.01	design	0.01
recognition	0.01	e	0.01	natural	0.01	engine	0.01
relations	0.01	page	0.01	cortex	0.01	service	0.01
library	0.01	gene	0.01	spike	0.01	social	0.01

Table 7.2: NetPLSA extracts cleaner topics

Topic 1		Topic 2		Topic 3		Topic 4	
retrieval	0.13	mining	0.11	neural	0.06	web	0.05
information	0.05	data	0.06	learning	0.02	services	0.03
document	0.03	discovery	0.03	networks	0.02	semantic	0.03
query	0.03	databases	0.02	recognit.	0.02	service	0.03
text	0.03	rules	0.02	analog	0.01	peer	0.02
search	0.03	association	0.02	vlsi	0.01	ontologi.	0.02
evaluation	0.02	patterns	0.02	neurons	0.01	rdf	0.02
user	0.02	frequent	0.01	gaussian	0.01	manage.	0.01
relevance	0.02	streams	0.01	network	0.01	ontology	0.01

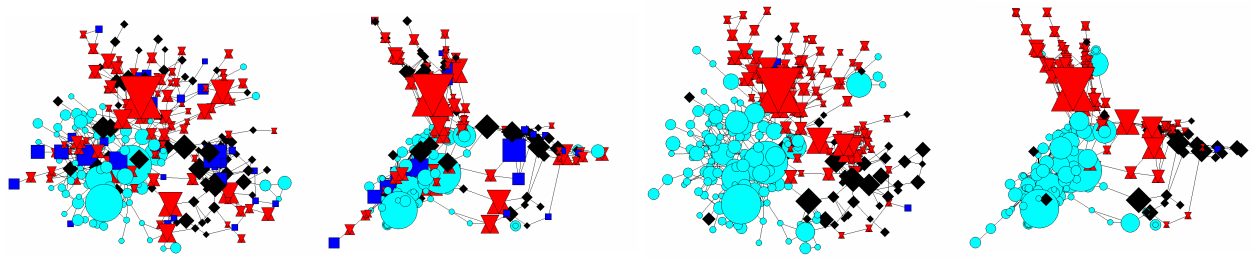
As a comparison with PLSA, we present the topics extracted with NetPLSA in Table 7.2. It is easy to see that the four topics regularized with the coauthor network are much cleaner. They are coherent enough to convey certain semantics, and general enough to cover the natural "topical communities". Specifically, Topic 1 well corresponds to the information retrieval (SIGIR) community, Topic 2 is closely related to the data mining (KDD) community, Topic 3 covers the machine learning (NIPS) community, and Topic 4 well covers the topic that is unique to the conference of WWW.

Topical Communities

We see that the quality of the topic models extracted with network regularization are better than those extracted without considering the network structure. However, could the regularized topic model really extract better topical communities? Are the topics in Table 7.2 really corresponding to coherent communities? We compare the topical communities identified by PLSA and NetPLSA.

Specifically, we assign each author to one of the topics, by $c_a = \arg\max_j p(\theta_j|a)$. We then visualize the authors assigned to different topics with different shapes and colors. The authors with the same shape thus form a topical community summarized by the corresponding topic model. As discussed in Section 7.2, the authors in the same topical community are expected to be well connected.

Figure 7.3 (a) and (b) present the topical communities extracted with the basic PLSA model, and Figure 7.3 (c) and (d) present the topical communities extracted with NetPLSA. With PLSA, although we can still see that lots of vertices in the same community are located closely, there aren't clear boundaries between communities. A considerable number of community members are scattered freely on the network (geometrically far from each other). On the other hand, when we regularize PLSA with the coauthor



(a) PLSA (b) PLSA: View II (c) NetPLSA (d) NetPLSA: View II

* Red: information retrieval; Black: data mining; Blue: machine learning; Grey: Web; $\lambda = 0.7$, $\gamma = 0.3$

Figure 7.3: Topical Communities in 4-CONF dataset

network, we see that the different communities can be identified clearly. Most authors assigned to the same topical community are well connected and closely located, which presents a much "smoother" pattern than Figure 7.3 (a) and (b).

Can we quantitatively prove that NetPLSA extracts better communities than PLSA? We have shown that network structure can help extracting topics. What about the reverse? Can a topic model of text help the network analysis?

Table 7.3: Quantitative Comparison of PLSA, NetPLSA, and Normalized Cut in Community Finding

Methods	Cut Edge weights	R. Cut/ N. Cut	Community Size ($\|V\|$)			
			C1*	C2	C3	C4
PLSA	4831	2.14/1.25	2280	2178	2326	2257
NetPLSA	**662**	0.29/0.13	2636	1989	3069	1347
NCut	855	**0.23/0.12**	2699	6323	**8**	**11**

*Ck means the k-th topical community, as in Table 7.1 and 7.2.

*Avg author weight: C1: 2.5; C2: 2.4; C3: 2.3; C4: **1.8**; All: 2.2

In Table 7.3, we quantitatively compare the performance of PLSA, NetPLSA, and a pure graph-based community extraction algorithm. We present the total weight of edges across different communities in column 2, and number of authors in each community in the rightmost 4 columns. Intuitively, if the communities are coherent, there should be many inner edges within each community and few cut edges across different communities. Clearly, there is significantly fewer cross community edges, and more inner community conductorships in the communities extracted by NetPLSA than PLSA. This means that NetPLSA indeed extracts more coherence topical communities than PLSA. Interestingly, we see that although Topic 4 (Web) in Table 7.2 is a coherent topic (more than 1300 authors are assigned to that topic), we cannot see a comparable number of members of this topical community from Figure 7.3, where we removed low degree authors and singletons (especially from Figure 7.3 (c) and (d)). This is because unlike IR, data mining and machine learning, "Web" is more an application field to the researchers than a focused research community, where many authors are from external communities and applying their techniques to the Web domain, and publishing papers to WWW. People purely assigned to the "WWW" topic either didn't publish many papers,

or are not well connected.

One may argue that in terms of inner/inter-community links alone, a community discovery algorithm which purely relies on the network structure may achieve a better performance. Indeed, what if we use the graph-based regularizer alone (by setting the $\lambda = 1$ in Equation 7.3 and including some constraints)? A quick answer maybe "*you can get intensively connected communities but you may not get semantically coherent ones*". To verify this, we compare our results with a pure graph-based clustering method. Specifically, we compare with the Normalized Cut (NC) clustering algorithm [157], which is one of the standard spectral clustering algorithms. By feeding the algorithm[4] with the coauthor matrix, we also extract four clusters (communities).

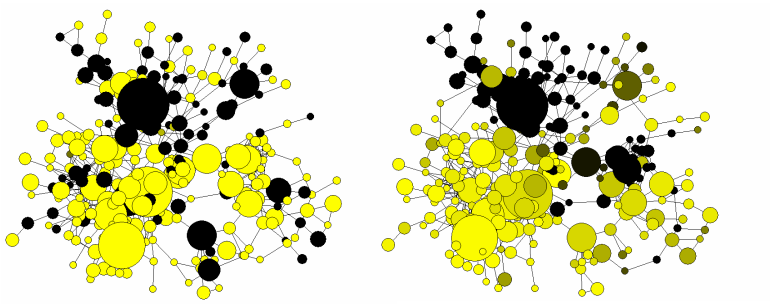
We present two other objectives of graph segmentation in the third column of Table 7.3, namely the "normalized cut [157]" and "ratio cut [17]". They respectively normalize the cross edges between two communities with the number of inner edges and the size of vertices in each community. If we solely consider the cut edges, it is hard to tell whether Normalized Cut or NetPLSA segments the network better, since one has a smaller "minimum cut" and the other has a smaller "normalized cut". However, in terms of topical communities, we see that our results are more reasonable. Community 3 and 4 of NC are extremely small. There is no way that they could represent a real research community, or any of the 4 conferences. Indeed, graph-based clustering algorithms are often trapped with small communities when the graph structure is highly skewed (or disconnected). In the network, we find that both cluster 3 and 4 are isolated components in the network (no out edges). They are both very coherent "*topological communities*", but not good *topical communities*, since the semantic information they represent is too narrow to cover a general topic. The involving of a topic model alleviates this sensitivity by bridging disconnected components with implicit topical correlations. This also guarantees semantical coherency within communities.

Even when a pure graph-based method extracts a perfect community, without the help of the topic model, it's hard to get a good topical summary of such a community. Community 1 of Normalized Cut well overlaps with the "information retrieval" community we got by NetPLSA. However, if we estimate a language model from the authors assigned to this community, we ends up with top probability words like "*web*", "*information*", "*retrieval*", "*neural*", "*learning*", "*search*", "*document*", etc (with MLE, and removing stop words). The semantics looks like a mixture and not as coherent as the NetPLSA results in Table 7.2. This is because in reality, an author usually works on more than one topics. Even when she is assigned to one community (even if assigned softly), we still need to exclude her work on other areas from the summary of this community, which cannot be achieved with just the network structure. This problem

[4] http://www.cis.upenn.edu/∼jshi/software/

is naturally solved with the involvement of a topic model, which assumes that a document covers multiple topics, and treats different words in a document differently.

Topic Mapping



(a) Topic Map with PLSA (b) With NetPLSA: Smoothed

Figure 7.4: Topic Map of "information retrieval" in 4-CONF dataset

Another basic task of TMN is to generate a map on the network for every topic. We use the probability $p(\theta|a)$ as the weighting function $f(v, \theta)$. We use the shades of a vertex to visualize the probability $p(\theta|a)$, where a darker vertex has a larger $p(\theta|a)$. As in Section 7.2, in a good topic map, the shades of adjacent vertices should be smooth.

In Figure 7.4, we visualize the topic map of "information retrieval" with the spring embedded view of the 4-CONF network. The darker a vertex is, the more likely the author belongs to the information retrieval community. From Figure 7.4 (a), we see that although a topic map could also be constructed with PLSA alone, the distribution of the topic on the network is desultory. It is hard to see where the topic origins, and how it is propagated on the network. We also see that PLSA likes to make extreme decisions, where an author is likely to be assigned an extremely large or small $p(\theta|a)$. In Figure 7.4 (b), however, we see that through the regularization with the coauthor network, the topic map is much smoother. We can easily identify the densest region of the topic "IR", and see it gradually propagates to the farther areas. Transitions between IR and non-IR communities are smooth, where the color of nodes changes from the darkest to the lightest in a gradational manner.

7.5.2 Geographic Topic Analysis

The other application we discussed in Section 7.4 is spatial topic analysis, more specifically, to model the geographic topic distributions. With this, we can analyze how a topic is propagating over the geographic locations. We create a collection of documents where each document is associated with a geographic location. All the geographic locations will then form a network structure based on their adjacency.

As discussed in [111, 114, 53], the weblog/blog data is a new genre of text data which is associated with rich demographic information. It is thus a suitable test bed for text mining problems with spatial analysis. Following [111], we collect weblog articles about a focused topic, by submitting a focused query to Google Blog Search[5], and crawling the content and geographic information of returned blog posts from their original websites. In this experiment, we use one of the data sets in [111], the Hurricane Katrina dataset. We also create a new dataset, with blog articles which contains the word "weather" in their titles. The basic statistics of the datasets are shown in Table 7.4.

Table 7.4: The basic statistics of blog datasets

Dataset	# docs	Time Span	Query
Katrina	4341*	8/16/05 - 10/04/05	"hurricane katrina"
Weather	493	10/01/06 - 9/30/07	"weather" in Title

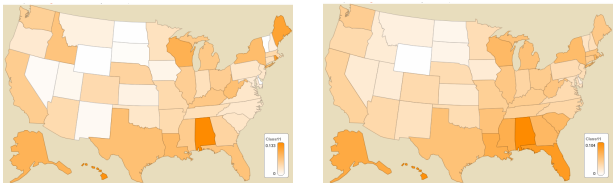
* Unlike [111], we only use the documents containing state labels. We restrict the domain in Live Spaces (http://spaces.live.com).

For both datasets, we create a vertex for every state in the U.S. and an edge between two adjacent states.

We use the model in Section 7.4.2 to extract topics with the context of geographic network structure. We then use the Many Eye visualization service[6] to visualize the spatial topic distribution of the one subtopic in hurricane Katrina. The subtopic discusses about the storms in Katrina, and in its successor hurricane Rita. Comparing Figure 7.5 (a) with Figure 7.5 (b), we see that the geographic distribution of topic is not dramatically different. This is reasonable, since the topic plotted in both figures is the same topic. However, we can still feel the difference between the figures: the topic distribution of Figure 7.5 (b) is much smoother than that in Figure 7.5 (a). Assume that a user does not know about hurricane Katrina or hurricane Rita, it is hard for her to guess where the events occurred from Figure 7.5 (a). People in Maine, Michigan, and Rhode Island seem to particularly focus on this topic, even more than people in Florida, Louisiana, and Mississippi (because of the sparsity of data in those states). From Figure 7.5 (b), however, we clearly see that the topic is densest along the Golf of Mexico, and gradually dilutes when it goes north and west. It is also clear that the discussion on this topic is denser in the west US than in the east. This is consistent with the reality, where the topic origins in the southeast coast, and gradually propagates to other states. In Figure 7.5 (b), we also see that the topic propagates smoothly between adjacent states. This also shows that our model could alleviate the overfitting problem of PLSA.

Let us show the results with another dataset, the Weather dataset in Table 7.4. Intuitively, when a user was discussing about weather in her blogs, the topics she chose to write about would be affected by where she lived. Since the topic "weather" is very broad, we guide the mixture model with some prior knowledge,

[5]http://blogsearch.google.com
[6]http://services.alphaworks.ibm.com/manyeyes/home



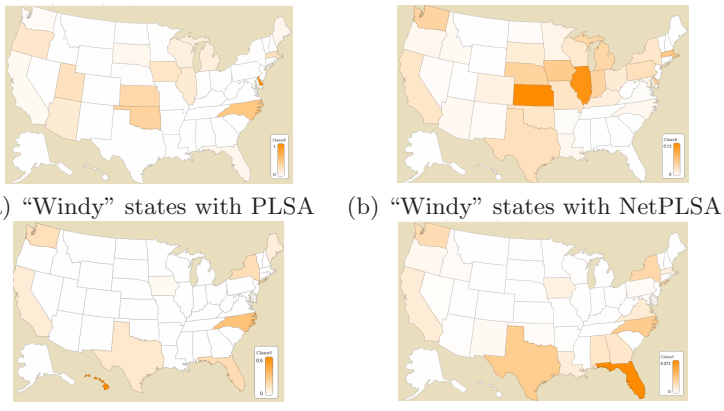
(a) With PLSA (b) With NetPLSA

Figure 7.5: Geographic Topic Distributions

so that it could extract several topics which we expect to see. Following [106], this is done by changing the MLE estimation of $p(w|\theta)$ in M step (Equation 7.11) into a maximum a posterior (MAP) estimation. We extract 7 topics from the Weather dataset. We use "wind" and "hurricane" as the prior for two of the topics, so that one of the output topic will be about the windy weather, and another will be about hurricanes. Table 7.5 compares the prior-guided topic models extracted from the Weather dataset. We see that with the network based regularizer, we indeed extract more coherent topics.

Table 7.5: Topic models: the Weather dataset

PLSA		NetPLSA	
"**wind**"	"**hurricane**"	"**wind**"	"**hurricane**"
windy	dean	windy	hurricanes
severe	storm	f	storms
pm	mexico	hi	tropical
thunderstorm	texas	cloudy	atlantic
hail	category	lo	season
watch	jamaica	lows	erin
blah	oil	highs	houston
probability	tourists	mph	louisiana



(a) "Windy" states with PLSA (b) "Windy" states with NetPLSA

(c) "Hurricane" with PLSA (d) "Hurricane" with NetPLSA

Figure 7.6: Geographic distribution of topics in Weather

In Figure 7.6, we visualize the geographic distributions of two weather topics over the US states. Comparing to the distributions computed with PLSA, we see that with NetPLSA, we can get much smoother distributions. PLSA assigns extremely large (close to 1) $p(\theta|d)$ of the topic "windy" to Delaware, and "hur-

ricane" to Hawaii. On the other hand, it assigns surprisingly low probability of "windy" to Texas. It also assigns extremely low probability of "hurricane" to Mississippi, Alabama and Georgia, although they are among the most vulnerable states to hurricanes. Through the regularization with states network, we see that this problem is alleviated. Northern midwest states and Texas are identified as windy states, especially Illinois (where the "windy city" Chicago locates). The southeast coasts, especially the states alone the Golf of Mexico (Florida as a representative), are identified as "hurricane" states.

In this section, we showed that with network based regularization, the NetPLSA model outperforms PLSA. It also extracts more robust topical communities than solely graph-based methods. NetPLSA generates coherent topics, topologically and semantically coherent communities, smoothed topic maps, and meaningful geographic topic distributions.

7.6 Related Work

Statistical topic modeling and social network analysis have little overlap in existing literature. Statistical topic modeling [61, 10, 184, 162, 111, 114, 90] uses a multinomial word distribution to represent a topic, and explains the generation of the text collection with a mixture of such topics. However, none of these existing models considers the natural network structure in the data. In the basic models such as PLSA [61] and LDA [10], there is no constraint other than "sum-to-one" on the topic-document distributions. [158] uses a regularizer based on KL divergence, by discouraging the topic distribution of a document from deviating the average topic distribution in the collection. We propose a different method, by regularizing a statistical topic model (e.g., PLSA) with the network structure associated with the data.

Contextual text mining [184, 162, 111, 114] is concerned with modeling topics and discovering other textual patterns with the consideration of contextual information, such as time, geographic location, and authorship. Our work is the first attempt where a network structure is considered as the context in topic models.

Social network analysis has been a hot topic for quite a few years. Many techniques have been proposed to discover communities [71, 3], model the evolution of the graph [88], and understand the diffusion of social networks [53, 89]. However, the rich textual information associated with the social network is ignored in most cases.

Although there has been some existing explorations [32, 104, 98, 5], there has not been a unified way to combine textual contents with social networks. Indeed, [187] proposes a probabilistic model to extract e-communities based on the content of communication documents, but they leave aside the network structure

in their model. Cohn and Hofmann proposed a model which combines PLSA and PHITS on the web graph [32]. Both topic and link are modeled as generated from a probabilistic mixture model. Their model, however, assumes a directed graph and does not directly optimize the smoothness of topics on the graph. To the best of our knowledge, combining topic modeling with graph-based harmonic regularization is a novel approach.

The graph-based regularizer is related to existing work in machine learning, especially graph-based semi-supervised learning [189, 185, 6, 188] and spectral clustering [17, 157]. The optimization framework we propose is closely related to [190], which is probably the first work combining a generative model with graph-based regularizer. Our work is different from theirs, as their task is semi-supervised classification, while we focus on unsupervised text mining problems such as topic modeling. NetSTM is a generalization of harmonic mixture to multiple topics and unsupervised learning.

The concrete applications we introduced in Section 7.4 are also related to existing work on author-topic analysis [162, 114], spatiotemporal text mining [111, 114], and blog mining [53, 111]. [186] explores co-author network to estimate the Markov transition probabilities between topics, which uses the network structure as a post processing step of topic modeling. Our work leverages the generative topic modeling and discriminative regularization in a unified framework.

7.7 Summary

In many knowledge discovery tasks, we encounter a data collection with both abundant textual information and a network structure. Statistical topic models extract coherent topics from the text, while usually ignoring the network structure. Social network analysis on the other hand, tends to focus on the topological network structure, while leaving aside the textual information. In this work, we formally define the major tasks of topic modeling with network structure. We propose a general solution of text mining with network structure, which optimizes the likelihood of topic generation and the topic smoothness on the graph in a unified way. Specifically, we propose a regularization framework for statistical topic models, with a harmonic regularizer based on the network structure. The general framework allows arbitrary choices of the topic model and the graph based regularizer. We show that with concrete choices, the model can be applied to tackle real world text mining problems such as author-topic analysis, topical community discovery, and spatial topic analysis.

Empirical experiments on two different genres of data show that our proposed method is effective to extract topics, discover topical communities, build topic maps, and model geographic topic distributions. It improves both pure topic modeling, and pure graph-based method.

This chapter serves as a general guideline of how to handle the dependency structures of context in contextual language models. Through the proposed graph-based regularizer, we can incorporate important assumptions and principles into the contextual language model, such as the third principle discussed in Section 3.4. This chapter gives a detailed discussion of the general component of contextual topic analysis discussed in Section 4.5.

The proposed graph-based regularizer provides a very general way to constrain the parameters of the contextual topic model, or a contextual language model in general. It can work along with all previous shown instantiations of the contextual mixture model, including the general contextual mixture model in Section 3.5.1, PLSA, CPLSA, and TSM. As long as there is a dependency structure of the contexts, we can leverage this regularizer to guarantee that similar contexts have similar context models.

Chapter 8

Postprocessing - Discovering Evolutionary Theme Patterns

In previous chapters, we introduced a very general conceptual framework of contextual text mining, a functional framework of contextual text mining with topics involved - contextual topic analysis, and specific instantiations of contextual topic analysis to handle explicit context, implicit context, and complex context beyond topics. These techniques provide detailed guidelines on how to construct special cases of the CPLSA model for specific contexts and specific tasks, how to add model priors to incorporate guidance from the user, and how to regularize the contextual topic model to capture the dependency structure of contexts. A variety of contextual topic patterns are extracted from these contextual topic models.

In this chapter and the next a few chapters, we will present some specific applications of contextual text mining. We especially focus on how to postprocess those basic contextual pattern extracted to facilitate real world text mining tasks. The contextual language models used in these applications are all special cases of the CPLSA models that we have discussed. The key problem is how to utilize the basic contextual patterns extracted from the contextual topic model, to extract refined contextual patterns, or to facilitate a variety of learning tasks. This chapter covers applications of discovering evolutionary theme patterns, where the main contexts considered are time and topics (see [113]). We can see that evolutionary theme patterns include different types of refined contextual patterns, which can not be inferred from the basic contextual patterns (i.e., the conditional distributions of contexts and language units) directly, but have to be extracted by postprocessing the basic contextual patterns.

8.1 Overview

In many application domains, we encounter a stream of text data, in which each text document has some meaningful time stamp. For example, a collection of news articles about a topic and research papers in a subject area can both be viewed as natural text streams where the articles are ordered according to their publication dates. In such stream text data, there often exist interesting temporal patterns. For example, an event covered in news articles generally has an underlying temporal and evolutionary structure consisting

of themes (i.e., subtopics) characterizing the beginning of the event, the progression of the event, and its impact, among others. Similarly, in research papers, research topics may also exhibit evolutionary patterns. For example, the study of one topic in some time period may have influenced or stimulated the study of another topic after the time period. In all these cases, it would be very useful if we can discover, extract, and summarize these evolutionary theme patterns (ETP) automatically. Indeed, such patterns not only are useful by themselves, but also would facilitate organization and navigation of the information stream according to the underlying thematic structures.

Consider, for example, the Asian tsunami disaster that happened in the end of 2004. A query to Google News (http://news.google.com) returned more than 80,000 online news articles about this event within one month (Jan.17 through Feb.17, 2005). It is generally very difficult to navigate through all these news articles. For someone who has not been keeping track of the event but wants to know about this disaster , a summary of this event would be extremely useful. Ideally, the summary would include both the major subtopics about the event and any threads corresponding to the evolution of these themes. For example, the themes may include the report of the happening of the event, the statistics of victims and damage, the aids from the world, and the lessons from the tsunami. A thread can indicate when each theme starts, reaches the peak, and breaks, as well as which subsequent themes it influences. A timeline-based theme structure as shown in Figure 8.1 would be a very informative summary of the event, which also facilitates navigation through themes.

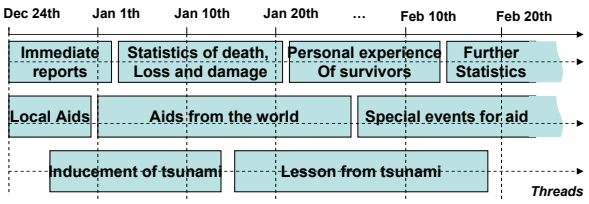


Figure 8.1: An example of theme thread structure

In addition to the theme structure, revealing the strength of a theme at different time periods, or the "life cycle" of a theme, is also very useful. Consider another scenario in the scientific literature domain. In a given research area, there are often hundreds of papers published annually. A researcher, especially a beginning researcher, often wants to understand how the research topics in the literature have been evolving. For example, if a researcher wants to know about information retrieval, both the historical milestones and the recent research trends of information retrieval would be valuable for him/her. A plot, such as the one shown in Figure 8.2, which visualizes the evolution patterns of research topics, would not only serve as a

good summary of the field, but also make it much easier for the researcher to selectively choose appropriate papers to read based on his/her research interests.

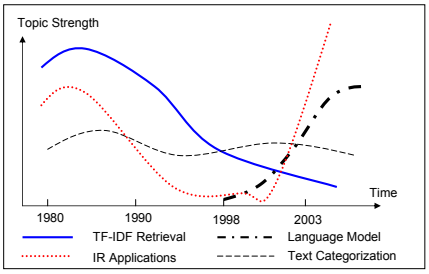


Figure 8.2: An example of theme strength in IR

In both scenarios, we clearly see a need for discovering evolutionary theme patterns in a text stream. In general, it is often very useful to discover the temporal patterns that may exist in a stream of text articles, a task which we refer to as *Temporal Text Mining* (TTM). Since most information bears some kinds of time stamps, TTM can be expected to have many applications in multiple domains.

Despite its importance, however, to the best of our knowledge, TTM has not been well addressed in the existing work. Most existing text mining work (e.g., [42, 56]) does not consider the temporal structures of text. Kleinberg's work on discovering bursty and hierarchical structures in streams [72] represents a major previous work on TTM, in which text streams are converted to temporal frequency data and an infinite-state automaton is used to model the stream. However, this method is inadequate for generating the evolutionary theme patterns as shown in the two examples above. A more detailed discussion of related work is given in Section 8.5.

In this chapter, we study the problem of discovering and summarizing the evolutionary patterns of themes in a text stream. We define this problem and present general probabilistic methods for solving the problem through (1) discovering latent themes from text, which includes both interesting global themes and salient local themes in a given time period; (2) discovering theme evolutionary relations and constructing an evolution graph of themes; and (3) modeling theme strength over time and analyzing the life cycles of themes. We evaluate the proposed methods on two data sets – a collection of 50 day's worth of news articles about the tsunami event (Dec.19, 2004 – Feb.08, 2005) and the abstracts of the ACM KDD conference papers from 1999 through 2004. The results show that our methods can discover many interesting evolutionary theme patterns from both data sets. In addition to news summarization and literature mining, the proposed temporal text mining methods are also directly applicable to many other application domains, including email analysis, mining user logs, mining customer reviews.

The rest of the chapter is organized as follows. In Section 8.2, we formally define the general problem of ETP discovery. In Section 8.3.1, we present our approaches to extracting themes and constructing a theme evolution graph. In Section 8.3.2, we further present a hidden Markov model based method for analyzing the life cycles of themes. We discuss our experiments and results in Section 8.4. Finally, Section 8.5 and Section 8.6 are related work and conclusions, respectively.

8.2 Problem Definition

The general problem of *evolutionary theme pattern* discovery can be formulated as follows.

Suppose we have a collection of time-indexed text documents, $\mathcal{D} = \{d_1, d_2, ..., d_T\}$, where d_i refers to a document with time stamp i. Each document is a sequence of words from a vocabulary set $V = \{w_1, ..., w_{|V|}\}$. Let us first define the following concepts.

Definition 8.1 (Theme Span) A ***theme span*** γ is defined as a theme θ spanning a given time interval l and is represented by $\langle \theta, s(\gamma), t(\gamma) \rangle$, where $s(\gamma)$ and $t(\gamma)$ are the starting and termination time stamps of l, respectively.

A theme span is a useful concept for associating time with themes. For the purpose of temporal text mining, a theme is almost always tagged with a time span. We thus use "theme" and "theme span" interchangeably whenever there is no ambiguity. We call a theme span that spans the entire text stream a ***trans-collection theme***. Thus if $\gamma = \langle \theta, s, t \rangle$ is a trans-collection theme, we must have $s = 1$ and $t = T$.

Definition 8.2 (Evolutionary Transition) Let Γ be a set of theme spans and $\gamma_1 = \langle \theta_1, s(\gamma_1), t(\gamma_1) \rangle$ and
$\gamma_2 = \langle \theta_2, s(\gamma_2), t(\gamma_2) \rangle \in \Gamma$ be two theme spans. If $t(\gamma_1) \leq s(\gamma_2)$ (γ_1 terminates before γ_2 starts) and the similarity between theme span γ_1 and γ_2 is above a give threshold, we say that there is an ***evolutionary transition*** from γ_1 to γ_2, which we denote by $\gamma_1 \prec \gamma_2$. We also say that θ_2 is evolved from θ_1, or θ_1 evolves into θ_2. We use $\mathcal{E} \subset \Gamma \times \Gamma$ to denote all the evolutionary transitions. That is, if $(\gamma_1, \gamma_2) \in \mathcal{E}$, then $\gamma_1 \prec \gamma_2$.

The concept of evolutionary transition is useful for describing the evolution relations between theme spans. With this concept, we can now define a particularly interesting theme pattern called a *theme evolution thread*.

Definition 8.3 (Theme Evolution Thread) Let Γ be a set of theme spans, a ***theme evolution thread*** is a sequence of theme spans $\gamma_0, \gamma_1, ..., \gamma_n \in \Gamma$ such that $(\gamma_i, \gamma_{i+1}) \in \mathcal{E}$.

Intuitively, a theme evolution thread characterizes how a family of related themes evolve over time. Since a text stream generally has multiple such theme threads, we now define another concept called *theme*

evolution graph to characterize the overall theme evolution patterns of a text stream.

Definition 8.4 (Theme Evolution Graph) A ***Theme Evolution Graph*** is a weighted directed graph $G = (N, E)$ in which each vertex $v \in N$ is a theme span, and each edge e $\in E$ is an evolutionary transition. The weight on an edge indicates the evolution distance. Clearly, each path in a theme evolution graph represents a theme evolution thread.

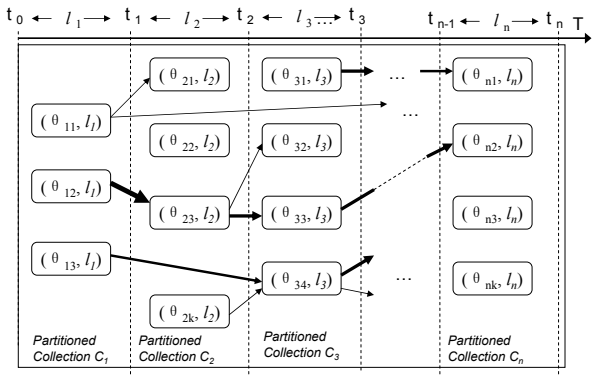


Figure 8.3: An example of a theme evolution graph

An example of a theme evolution graph is shown in Figure 8.3, where each vertex is a theme span extracted from a subcollection obtained through non-overlapping partitioning of the stream into n sliced intervals. Each edge is an evolutionary transition. The thickness of an edge indicates how close the two themes being connected are and how trustful the corresponding evolutionary transition is; a thicker edge indicates a closer distance between the themes and a more trustful transition. For example, the distance of $\theta_{12} \prec \theta_{23}$ is smaller than that of $\theta_{11} \prec \theta_{21}$, and the former is more trustful. We also see a theme evolution thread from θ_{12}, through θ_{23}, and all the way to θ_{n2}.

The theme evolution graph discussed above gives us a microcosmic view of the ETPs – revealing the major theme spans within each time interval and their evolutionary structures. To obtain a macroscopical view of the ETPs, it would be useful to extract the global evolutionary patterns of themes over the whole text stream and analyze the "life cycle" of each specific theme.

Definition 8.5 (Theme Life Cycle) Given a text collection tagged with time stamps and a set of trans-collection themes, we define the ***Theme Life Cycle*** of each theme as the strength distribution of the theme over the entire time line. The strength of a theme at each time period is measured by the number of words generated by this theme in the documents corresponding to this time period, normalized by either the number of time points (giving an *absolute strength*), or the total number of words in the period (giving a *relative strength*). The absolute strength measures the absolute amount of text which a theme can explain,

while the relative strength indicates which theme is relatively stronger in a time period.

Given a text stream $\mathcal{D}$, a major task of the general **Evolutionary Theme Pattern discovery** problem is to extract a theme evolution graph from $\mathcal{D}$ automatically. Such a graph can immediately be used as a summary of the themes and their evolution relations in the text stream, and can also be exploited to organize the text stream in a meaningful way. Sometimes, a user may be interested in a specific theme. For example, a researcher may be interested in a particular subtopic. In this case, it is often useful to analyze the whole "life cycle" of a theme thread. Thus another task of ETP discovery is to compute the strength of a theme at different time periods so that we can see when the theme has started, when it is terminated, and whether there is any break in between.

One can easily see that the problem of the ETP discovery is an instantiation of contextual text mining, specifically an instantiation of the contextual topic analysis. The contexts involved in this task are global themes, local theme spans, and time periods. We can apply a contextual topic model to deal with this problem. However, some of the major evolutionary theme patterns, such as the theme transitions, the evolutionary theme graph, the theme threads, can not be directly modeled as conditional distributions of contexts and language units. Instead, we need a way to extract these refined patterns by postprocessing the basic contextual topic patterns.

The ETP discovery problem is challenging in many ways. First, it is a completely unsupervised task; there's no training data to discriminate theme spans. This indicates a great advantage of any techniques for ETP discovery – no/minimum prior knowledge about a domain is assumed. Second, unlike the problem of novelty detection and event tracking, which aims to segment the text and find the boundaries of events [13, 144, 99], the ETP discovery problem involves the more challenging task of modeling the multiple subtopics at any time interval for an event, and aims to discover the changing and evolutionary relations between the theme spans. Finally, the analysis of theme life cycles requires the system to decode the whole collection with themes and model the strength variations of each theme along the time line in a completely unsupervised way.

8.3 Methods

8.3.1 Evolutionary Theme Graphs

Given a stream of text $\mathcal{D} = \{d_1, d_2, ..., d_T\}$, our goal is to extract a theme evolution graph from $\mathcal{D}$ automatically. At a high-level, our methods involve the following three steps:

1. Partition the documents into n possibly overlapping subcollections with fixed or variable time intervals

so that $\mathcal{D} = \mathcal{D}_1 \cup ... \cup \mathcal{D}_n$ and $\mathcal{D}_i = \{d_{t_i}, ..., d_{t_i+l_i-1}\}$ is a subcollection of l_i documents in the time span $[t_i, t_i + l_i - 1]$. In general, $t_i < t_{i+1}$, but it may be that $t_i + l_i - 1 > t_{i+1}$, since $\mathcal{D}_i$'s may be overlapping. The actual choice of the interval lengths l_i and whether $\mathcal{D}_i$'s should overlap are determined by specific applications.

2. Extract the most salient themes $\Theta_i = \{\theta_{i,1}, ..., \theta_{i,k_i}\}$ from each subcollection $\mathcal{D}_i$ using a probabilistic mixture model.

3. For any themes in two different subcollections, $\theta_1 \in \Theta_i$ and $\theta_2 \in \Theta_j$ where $i < j$, decide whether there is an evolutionary transition based on the similarity of θ_1 and θ_2.

The first step is trivial; the second step can be easily conducted by applying the PLSA model directly on each $\mathcal{D}_i$; we describe the last step in detail. Note that the third step serves as a postprocessing step for contextual text mining, by inferring refined contextual patterns (i.e., theme evolution transitions and evolutionary theme graph) from the basic contextual patterns.

With the theme spans extracted from all the subcollections, we now turn to the discovery of evolutionary transitions. To discover any evolutionary transition between two theme spans, we use the Kullback-Leibler divergence [34] to measure their evolution distance. Let $\gamma_1 = \langle \theta_1, s(\gamma_1), t(\gamma_1) \rangle$ and $\gamma_2 = \langle \theta_2, s(\gamma_2), t(\gamma_2) \rangle$ be two theme spans where $t(\gamma_1) \leq s(\gamma_2)$. We assume that γ_2 has a smaller evolution distance to γ_1 if their unigram language models θ_2 and θ_1 are closer to each other. Since the KL-divergence $D(\theta_2||\theta_1)$ can model the additional new information in θ_2 as compared to θ_1, it appears to be a natural measure of evolution distance between two themes.

$$D(\theta_2||\theta_1) = \sum_{i=1}^{|V|} p(w_i|\theta_2) \log \frac{p(w_i|\theta_2)}{p(w_i|\theta_1)}$$

Note that the KL-divergence is asymmetric and it makes more sense to use $D(\theta_2||\theta_1)$ than $D(\theta_1||\theta_2)$ to measure the evolution distance from θ_1 to θ_2.

For every pair of theme spans γ_1 and γ_2 where $t(\gamma_1) \leq s(\gamma_2)$, we compute $D(\theta_2||\theta_1)$. If $D(\theta_2||\theta_1)$ is above a threshold ξ, we will infer that $\gamma_1 \prec \gamma_2$. The threshold ξ allows a user to flexibly control the strength of the theme transitions.

Once we extract the theme spans from all the subcollections and identify all the evolutionary transitions, we essentially have a theme evolution graph.

8.3.2 Theme Life Cycles

Note the the definition of theme life cycle in this chapter is different from the definition of the topic life cycle in Chapter 5, but similar the the definition of the sentiment dymanics in Chapter 6. The basic idea is that the strength of a topic is computed using the actual number of words in the active domain of a topic, instead of using the conditional distribution $p(t|\theta)$ directly. The advantage of doing this is that the strength of different topics across different time periods are all comparable. The downside, however, is that this strength has to be inferred by postprocessing the basic topic patterns instead of use the conditional distributions directly.

We now present a method based on Hidden Markov Models (HMMs) [141] to model and decode the shift between trans-collection themes in the whole collection. Based on the decoding results, we can then compute the theme strengths and analyze theme life cycles in a straightforward way.

We first give a brief introduction to HMMs. An HMM can be characterized by a set of hidden states $S = \{s_1, ..., s_n\}$, a set of observable output symbols $O = \{o_1, ..., o_m\}$, an initial state probability distribution $\{\pi_i\}_{i=1}^n$, a state transition probability distribution $\{a_{i,j}\}_{j=1}^n$ for each state s_i, and an output probability distribution $\{b_{i,k}\}_{k=1}^m$ for each state s_i. An HMM defines a generative probabilistic model for any sequence of symbols from O with parameters satisfying the following constraints: (1) $\sum_{i=1}^n \pi_i = 1$; (2) $\sum_{j=1}^n a_{i,j} = 1$; (3) $\sum_{k=1}^m b_{i,k} = 1$.

To use an HMM to model the stochastic process of theme shifts in our text stream, we assume that the collection, which is represented as a long sequence of words, is stochastically generated from a sequence of unobservable theme models, where the shifts between themes are represented as the transitions between the states of an HMM that correspond to the extracted theme models. Such an HMM is constructed in the following way. We first extract k trans-collection themes from the collection. We then construct a fully connected HMM with $k + 1$ states, of which k states correspond to the extracted k themes and the other one corresponds to a background theme language model estimated based on the whole collection. The entire vocabulary V is taken as the output symbol set, and the output probability distribution of each state is set to the multinomial distribution of words given by the corresponding theme language model.

An example of a 3-theme HMM is shown in Figure 8.4. The background state, which corresponds to the background theme model, aims to account for non-discriminative words, while the content words and subtopics are modeled by the states corresponding to the trans-collection themes. Since the extracted themes are discriminative, we may reasonably assume that each theme can only shift to another theme through the background model. The unknown parameter set in the HMM is $\Lambda = \{\pi_i, a_{i,i}, a_{i,B}, a_{B,i}\}_{i=1}^n$. Λ can be

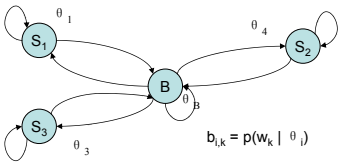


Figure 8.4: A 3-theme transition HMM structure

estimated using an EM algorithm called Baum-Welch algorithm [141].

Once the initial state probabilities and transition probabilities are estimated, the Viterbi algorithm [141] can be used to decode the text stream to obtain the most likely state sequence, i.e., the most likely sequence of theme shifts, as shown in Figure 8.5.

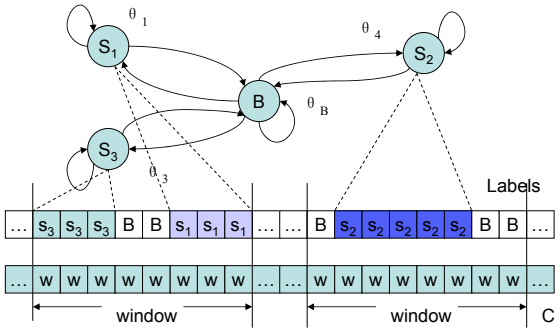


Figure 8.5: Decoding the collection

Once the whole stream $C = \{d_1, ..., d_T\}$ is decoded with the labels of themes, we can use a fixed-size sliding window of time to measure the strength of each theme at a time point[1]. Let $d_i = d_{i1}...d_{i|d_i|}$ be the sequence of words in d_i. The absolute and relative strengths of theme i at time t is computed as:

$$AStrength(i,t) = \frac{1}{W} \sum_{t' \in [t-\frac{W}{2}, t+\frac{W}{2}]} \sum_{j=1}^{|d_{t'}|} \delta(d_{t'j}, i)$$

where $\delta(d_{t'j}, i) = 1$ if word $d_{t'j}$ is labeled as theme i; otherwise $\delta(d_{t'j}, i) = 0$. W is the size of the sliding

[1]The use of a sliding window also avoids the "report delay" problem in the news domain.

window in terms of time points.

$$\begin{aligned} NStrength(i,t) &= \frac{AStrength(i,t)}{\sum_{j=1}^{k} AStrength(j,t)} \\ &= \frac{\sum_{t' \in [t-\frac{W}{2}, t+\frac{W}{2}]} \sum_{j=1}^{|d_{t'}|} \delta(d_{t'j}, i)}{\sum_{t' \in [t-\frac{W}{2}, t+\frac{W}{2}]} |d_{t'}|} \end{aligned}$$

The life cycle of each theme can then be modeled as the variation of the theme strengths over time.

The analysis of theme life cycles thus involves the following four steps: (1) Construct an HMM to model how themes shift between each other in the collection. (2) Estimate the unknown parameters of the HMM using the whole stream collection as observed example sequence. (3) Decode the collection and label each word with the hidden theme model from which it is generated. (4) For each trans-collection theme, analyze when it starts, when it terminates, and how it varies over time.

8.4 Experiments

We evaluated the proposed ETP discovery methods on two different domains - news articles and scientific literature.

8.4.1 Data Preparation

Two data sets are constructed to test our approaches. The first, tsunami news data, consists of news articles about the event of Asia Tsunami dated Dec. 19 2004 to Feb. 8 2005. We downloaded 7468 news articles from 10 selected sources, with the keyword query "tsunami". As shown in Table 8.1, three of the sources are in Asia, two of them are in Europe and the rest are in the U.S.

News Source	Nation	News Source	Nation
BBC	UK	Times of India	India
CNN	US	VOA	US
Economics Times	India	Washington Post	US
New York Times	US	Washington Times	US
Reuters	UK	Xinhua News	China

Table 8.1: News sources of Asia Tsunami data set

The second data set consists of the abstracts in KDD conference proceedings from 1999 to 2004. All the abstracts were extracted from the full-text pdf files downloaded from the ACM digital library[2]. Some documents which were not recognizable by the pdf2text software in Linux were excluded. The basic statistics

[2]http://www.acm.org/dl

of Tsunami news data and KDD Abstract data are shown in Table 8.2. We intentionally did not perform stemming or stop word pruning in order to test the robustness of our algorithms.

Data Set	# of docs	AvgLength	Time range
Asia Tsunami	7468	505.24	12/19/04 - 02/08/05
KDD Abs.	496	169.50	1999-2004

Table 8.2: Basic information of data sets

On each data set, two major experiments are designed: (1) Partition the collection into time intervals, discover the theme evolution graph and identify theme evolution threads. (2) Discover trans-collection themes and analyze their life cycles. The results are discussed below.

	Theme 1	Theme 2	Theme 3	Theme 4	Theme 5	Theme 6
l1:	system 0.0104	Year 0.0074	debt 0.0148	Aceh 0.0320	Annan 0.0081	match 0.0094
Dec/	Bush 0.0080	silence 0.0056	Club 0.0098	Indonesia 0.0118	U.N. 0.0064	XI 0.0065
19/	warning 0.0070	British 0.0053	Paris 0.0097	military 0.0118	summit 0.0062	players 0.0059
04	dollars 0.0067	New 0.0051	Bank 0.0063	Banda 0.0096	children 0.0060	Cricket 0.0058
	million 0.0064	celebrations 0.0050	moratorium 0.0061	Indonesian 0.0089	Powell 0.0044	game 0.0050
-	small 0.0058	UK 0.0047	freeze 0.0058	province 0.0088	NBC 0.0037	Zealand 0.0044
	US 0.0055	music 0.0038	repayments 0.0052	workers 0.0087	million 0.0036	Australia 0.0042
Jan/	conference 0.0052	London 0.0038	billion 0.0044	foreign 0.0081	disease 0.0035	Sudan 0.0039
04/	meeting 0.0035	Sydney 0.0037	U.N. 0.0044	islands 0.0077	WHO 0.0033	captain 0.0038
05	Egeland 0.0033	Blair 0.0035	nations 0.0042	aid 0.0071	UNICEF 0.0031	Ponting 0.0036
l2:	countries 0.0240	Mr 0.0104	Aceh 0.0226	missing 0.0143	her 0.0147	match 0.0075
Dec/	debt 0.0146	Blair 0.0068	aid 0.0204	Thailand 0.0115	islands 0.0102	Cricket 0.0065
28/	system 0.0085	British 0.0062	Powell 0.0171	bodies 0.0107	Nicobar 0.0098	players 0.0052
04	nations 0.0084	Rs 0.0057	relief 0.0161	dead 0.0071	I 0.0069	XI 0.0052
	China 0.0073	Britons 0.0047	Indonesia 0.0160	Sweden 0.0068	she 0.0067	you 0.0046
-	warning 0.0064	UK 0.0046	Annan 0.0134	Thai 0.0065	Andaman 0.0064	Zealand 0.0042
	Paris 0.0064	donations 0.0045	U.S. 0.0131	Swedish 0.0064	beach 0.0064	game 0.0033
Jan/	Club 0.0058	crore 0.0037	United 0.0122	police 0.0060	sea 0.0064	points 0.0033
14/	Bank 0.0056	Tamil 0.0036	military 0.0113	DNA 0.0056	my 0.0060	captain 0.0032
05	Chinese 0.0054	public 0.0033	U.N. 0.0110	tourists 0.0052	island 0.0051	cricket 0.0032
l3:	Chinese 0.0085	Tamil 0.0121	toll 0.0103	warning 0.0121	United 0.0228	Thailand 0.0103
Jan/	British 0.0076	Sri 0.0121	bodies 0.0083	system 0.0119	Powell 0.0168	missing 0.0092
05/	UK 0.0075	Lanka 0.0070	death 0.0067	islands 0.0086	Bush 0.0165	Phuket 0.0087
05	China 0.0070	Nadu 0.0061	dead 0.0067	sea 0.0061	U.S. 0.0146	Khao 0.0070
	Hong 0.0068	Tigers 0.0059	debt 0.0063	Nicobar 0.0048	States 0.0137	her 0.0070
-	Kong 0.0064	government 0.0050	food 0.0057	Pacific 0.0047	Mr. 0.0117	beach 0.0068
	donations 0.0060	Lankan 0.0040	Paris 0.0057	water 0.0042	Nations 0.0101	Lak 0.0067
Jan/	Red 0.0056	Nicobar 0.0040	Indonesia 0.0056	Japan 0.0040	$ 0.0088	Swedish 0.0066
22/	concert 0.0052	Singh 0.0037	Club 0.0053	Kobe 0.0037	relief 0.0079	Sweden 0.0064
05	Cross 0.0050	rebels 0.0031	corpses 0.0051	quake 0.0033	million 0.0076	hotel 0.0059
l4:	Aceh 0.0250	funding 0.0046	Phi 0.0052	concert 0.0107	LTTE 0.0055	Iraq 0.0087
Jan/	talks 0.0175	Iraq 0.0044	her 0.0048	Kobe 0.0050	Tamil 0.0052	Bush 0.0086
15/	GAM 0.0150	Eid 0.0039	ASEAN 0.0036	singer 0.0045	talks 0.0037	billion 0.0084
05	rebels 0.0133	regional 0.0035	resort 0.0024	stars 0.0041	local 0.0036	pilgrims 0.0073
	peace 0.0100	festival 0.0034	Palu 0.0023	Stadium 0.0040	UK 0.0034	budget 0.0067
-	Indonesian 0.0085	congressional 0.0033	Palu 0.0023	Wales 0.0040	Tigers 0.0033	deficit 0.0060
	province 0.0074	mosque 0.0033	cancer 0.0022	Japan 0.0036	Hafun 0.0030	House 0.0059
Jan/	Free 0.0055	Rice 0.0032	Phuket 0.0021	rock 0.0035	Norwegian 0.0030	boat 0.0053
30/	Movement 0.0052	month 0.0030	Hui 0.0021	Millennium 0.0034	Prabhakaran 0.0029	Trump 0.0042
05	rebel 0.0048	military 0.0029	Fleming 0.0020	Live 0.0030	Kalpakkam 0.0028	spending 0.0042
l5:	Jones 0.0051	billion 0.0197	boat 0.0081	Clinton 0.0115	debt 0.0195	talks 0.0263
Jan/	Palu 0.0046	$ 0.0153	tourism 0.0067	var 0.0052	meeting 0.0136	Aceh 0.0213
23/	station 0.0045	Iraq 0.0140	Samui 0.0059	Nepal 0.0049	finance 0.0122	peace 0.0147
05	Pierson 0.0042	House 0.0121	ASEAN 0.0055	summit 0.0044	Brown 0.0087	Indonesian 0.0113
	song 0.0034	budget 0.0101	JAL 0.0054	SAARC 0.0042	exchange 0.0074	rebels 0.0112
-	North 0.0033	request 0.0094	tourists 0.0046	Dhaka 0.0036	ministers 0.0067	Helsinki 0.0094
	Korea 0.0033	funding 0.0086	accident 0.0041	construction 0.0030	agreed 0.0065	conflict 0.0077
Feb/	Miss 0.0031	White 0.0083	month 0.0041	Bangladesh 0.0026	gold 0.0054	province 0.0070
8/	97 0.0030	Afghanistan 0.0071	joke 0.0038	envoy 0.0025	IMF 0.0054	sides
05	show 0.0030	baby 0.0066	Marsh 0.0035	techniques 0.0021	economic 0.0047	autonomy

Table 8.3: Theme spans extracted from Asia Tsunami data

8.4.2 Experiments on Asia Tsunami

Since news reports on the same topic may appear earlier in one source but later in another (i.e., "report delay"), partitioning news articles into *overlapping*, as opposed to non-overlapping subcollections seems to be more reasonable. We thus partition the our news data into 5 time intervals, each of which spans about

two weeks and is half overlapping with the previous one. We use the mixture model discussed in Section 8.3.1 to extract the most salient themes in each time interval. We set the background parameter $\lambda_B = 0.95$ and number of themes in each time interval to be 6. The variation of λ_B is discussed later. Table 8.3 shows the top 10 words with the highest probabilities in each theme span. We see that most of these themes suggest meaningful subtopics in the context of the Asia tsunami event.

Source	Theme 1	Theme 2	Theme 3	Theme 4	Theme 5
CNN	system 0.0079	I 0.0322	children 0.0119	Aceh 0.0088	Bush 0.0201
	warning 0.0075	wave 0.0061	debt 0.0072	Indonesia 0.0063	$ 0.0173
	Ocean 0.0073	beach 0.0056	hospital 0.0072	said 0.0054	million 0.0135
	Indian 0.0064	water 0.0051	baby 0.0064	military 0.0044	relief 0.0134
	Pacific 0.0063	when 0.0050	Club 0.0063	U.N. 0.0038	United 0.0105
	earthquake 0.0061	saw 0.0046	Paris 0.0061	number 0.0032	aid 0.0099
	quake 0.0057	sea 0.0046	child 0.0054	survivors 0.0032	Powell 0.0098
	tsunami 0.0054	Thailand 0.0042	her 0.0053	reported 0.0031	U.S. 0.0075
	ocean 0.0039	family 0.0039	police 0.0048	helicopters 0.0028	States 0.0075
	scientists 0.0031	ran 0.0033	moratorium 0.0046	killed 0.0027	U.N. 0.0056
XINHUA	Thailand 0.0104	Aceh 0.0219	Chinese 0.0391	dollars 0.0226	system 0.0314
	Thai 0.0096	province 0.0111	China 0.0391	million 0.0204	warning 0.0272
	missing 0.0079	Indonesian 0.0075	yuan 0.0180	US 0.0178	early 0.0172
	victims 0.0054	tidal 0.0055	countries 0.0098	aid 0.0118	meeting 0.0159
	Philippine 0.0040	waves 0.0047	Beijing 0.0089	United 0.0108	Ocean 0.0121
	confirmed 0.0040	killed 0.0045	travel 0.0061	countries 0.0106	small 0.0096
	residents 0.0037	quake 0.0043	$ 0.0058	UN 0.0102	international 0.0092
	tourists 0.0033	island 0.0043	donated 0.0057	Annan 0.0082	conference 0.0086
	percent 0.0032	dead 0.0041	Cross 0.0053	debt 0.0071	natural 0.0082
	number 0.0032	death 0.0041	donation 0.0052	reconstruction 0.0062	disasters 0.0070

Table 8.4: Trans-collection themes extracted from CNN and Xinhua News

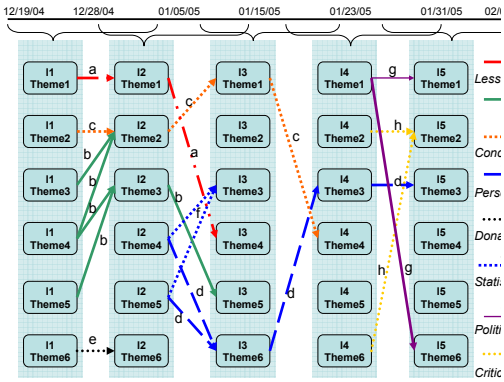


Figure 8.6: Theme evolution graph for Asia tsunami

With these theme spans, we use the KL-divergence to identify evolutionary transitions. Figure 8.6 shows a theme evolution graph discovered from Asia Tsunami data when the threshold for evolution distance is set to $\xi = 12$. From Figure 8.6, we can see several interesting evolution threads which are annotated with symbols.

The thread labeled with a may be about warning systems for tsunami (both "warning" and "system" are in each theme span of this thread). It is interesting to see that the nation covered by the thread seems to have evolved from the U.S. in period l_1, to China in l_2, and then to Japan in l_3. In another group of edges labeled with b, themes 3, 4, and 5 in period l_1 indicate the aids and financial support from UN, the

aids from local area, and special aids for children, respectively. All of them show an evolutionary relation to theme 2 (donation from UK) and theme 3 (aid from US) in l_2. The latter theme further evolves into theme 5 in l_3, which is mainly about money support from US. Another interesting evolutionary thread (thread c) begins with music-related events and aids from UK. It shifts to talk about concerts in Hong Kong and then Japan with the purpose of raising funds for donation. Note that although theme 3 and theme 4 in l_1 also show high proximity to theme 2 in l_2, we do not consider them as a part of thread c, because the aspect in which they are similar to theme 2 in l_2 is different to the semantic aspect of the global path c. Thread d is about the personal experiences of survivors. It starts with theme 5 in l_2 and goes through theme 6 in l_3, theme 3 in l_4. It finally evolves into theme 3 in l_5. There are also several short but noticeable theme evolution threads. For example, thread e is about cricket matches for donation, while thread f is about deaths and losses in the disaster.

It is interesting to see that, in the latest two time intervals, most themes are no longer about the tsunami event, indicating that the event has probably been receiving diminishing attention in these two periods, which will be seen more clearly when we look at the life cycles of themes later. Indeed, there are two politics-related short theme threads (i.e., g and h). In threads g, theme 1 in l_4 is about political issues ("rebels" and "peace"). It splits into two themes in l_5, about North Korea and the Aceh peace talk, respectively. Theme 2 and theme 5 in l_4 represent criticisms on the Iraq affair (one for military issues and one for the high expenditure/cost). In l_5, they merged into a single theme, which mentions the budget on Iraq issues and Afghanistan issues. Interestingly, by linking back to the articles, it turns out to be arguing for shrinking the budget on war issues and offering more aid for the disaster.

The second experiment aims to model the patterns of life cycles of trans-collection themes. In this experiment, we analyze the life cycles of trans-collection themes in two individual sources (CNN and Xinhua News) instead of the whole mixed collection to avoid "report delay". The five trans-collection themes extracted from CNN and Xinhua News are shown in Table 8.4.

The five themes from CNN roughly correspond to (1) research and lessons from Tsunami; (2) personal experience of survivors; (3) Special aid program for children; (4) general reports and statistics; (5) aids and donations from the world, especially from U.S. The five themes from Xinhua roughly correspond to (1) statistics of death and missing; (2) reports and stories at the scene; (3) donations from China; (4) aids and donations from the world; (5) research and lessons from Tsunami. CNN-theme1 and XINHUA-theme5; CNN-theme4 and XINHUA-theme1; CNN-theme5 and XINHUA-theme4 are strongly correlated common themes. CNN-theme2 and XINHUA-theme2 are correlated in a weaker sense. XINHUA-theme3 and CNN-theme5 clearly represent the different regionality of the two sources.

In Figure 8.7 we show the life cycles of the trans-collection themes in CNN by plotting the absolute

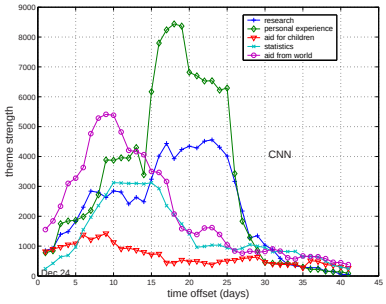


Figure 8.7: Theme life cycles in CNN data ($W = 10$)

theme strength values over time. We see that the absolute strength of all five themes are increasing in the first 10 days after Dec. 24, 2004. Reports on aids for children and aids from the world begin to decay all the time after that. General reports and statistics starts to decay around Jan 10 for the rest of the time. Around Jan. 7th, the reports on the research and lessons from tsunami start to increase again. The same pattern is discovered in reports on personal experiences. This is probably because survivors have come back to their home country. Both themes drop sharply around Jan. 17. After Jan. 22, all 5 themes retain a low strength level, indicating the event is receiving diminishing attention. The normalized strengths of themes in the CNN data show similar patterns.

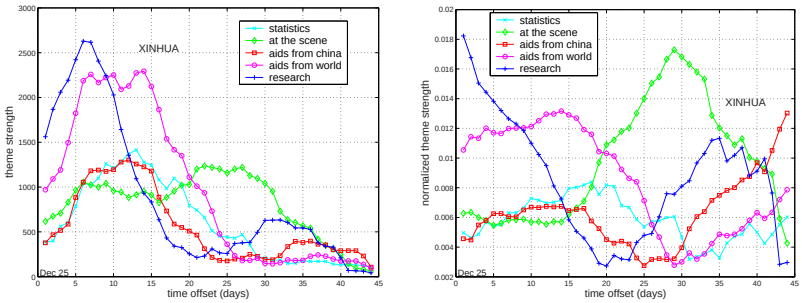


Figure 8.8: Theme life cycles in Xinhua News ($W = 10$)

In the two plots in Figure 8.8, we show the absolute and normalized theme strength values of the trans-collection themes over time in Xinghua News. We see that, in the first week beginning Dec. 25th 2004, all 5 themes are increasing rapidly. All themes begin to decay around Jan. 10th except for stories and reports at the scene, which increase again after a roughly 10-day period of mild decreasing. The theme about death statistics begins to decay all the time after Jan. 16. Both aids from China and research and lessons from tsunami present a second increasing at late January, although not as significant as the first one. In the normalized strength plot, it is easy to see that before Jan 3rd, the dominating theme is theme 5. In the next 10 days, aid from the world is most significant. In the following 20 days, "on scene stories" is the dominating

	1999	2000	2001	2002	2003	2004
THEME1	association 0.0156	ABSTRACT 0.0141	rules 0.0148	web 0.0089	Clustering 0.0077	topic 0.0104
	rules 0.0149	revision 0.0121	datasets 0.0104	hierarchical 0.0081	indices 0.0077	Algorithms 0.0103
	associations 0.0128	clustering 0.0120	artificial 0.0086	classification 0.0068	Recognition 0.0070	correlation 0.0086
	rule 0.0090	test 0.0106	support 0.0070	features 0.0057	mixture 0.00654	image 0.0079
	attribute 0.0062	die 0.0094	rule 0.0068	analysis 0.0055	Pattern 0.0065	mixture 0.0076
	dimension 0.0060	Terms 0.0093	SVM 0.0067	Markov 0.0053	random 0.0065	LDA 0.0060
	median 0.0057	classi 0.0090	criteria 0.0063	systems 0.0050	cluster 0.0060	metrics 0.0055
	attributes 0.0056	protein 0.0077	classification 0.0062	topics 0.0046	components 0.0055	spatial 0.0048
	polish 0.0050	control 0.0076	linear 0.0061	classification. 0.0046	clustering 0.0053	semantic 0.0048
	transaction 0.0047	functional 0.0060	useful 0.0061	intrusion 0.0045	variables 0.0052	incremental 0.0048
THEME2	algorithms 0.0155	rules 0.0192	disconnected 0.0102	gene 0.0164	Information 0.0122	genes 0.0184
	Abstract 0.0084	function 0.0116	web 0.0097	time 0.0153	cube 0.0117	problem 0.0100
	EPs 0.0077	sequence 0.0091	models 0.0092	series 0.0139	Web 0.0096	graph 0.0099
	objects 0.0059	set 0.0071	components 0.0083	change 0.0106	social 0.0075	structure 0.0088
	distance 0.0054	minimality 0.0067	graph 0.0082	statistical 0.0072	weighted 0.0066	Algorithms 0.0076
	clustering 0.0052	discovered 0.0064	boosting 0.0070	may 0.0069	Retrieval 0.0065	collection 0.0071
	sampling 0.0050	protein 0.0063	dimensionality 0.0068	detection 0.0065	distance 0.0059	subset 0.0068
	problem 0.0048	minimal 0.0061	random 0.0065	source 0.0063	user 0.0059	microarray 0.0060
	Bayesian 0.0046	functional 0.0060	reduction 0.0056	kernel 0.0062	Search 0.0054	semantic 0.0059
	profiles 0.0044	tuberculosis 0.0056	outlier 0.0052	base 0.0061	networks 0.0047	samples 0.0057
THEME3	Abstract 0.0104	students 0.0110	item 0.0117	classification 0.0150	Database 0.0102	part 0.0120
	itemsets 0.0085	level 0.0088	database 0.0096	algorithm 0.0130	data 0.0091	reviews 0.0083
	rules 0.0084	aggregate 0.0083	sets 0.0091	text 0.0128	expression 0.0082	BOM 0.0076
	local 0.0068	statistical 0.0081	frequent 0.0065	unlabeled 0.0113	gene 0.0074	opinion 0.0070
	tree 0.0067	statistics 0.0077	compounds 0.0059	document 0.0087	results 0.0069	maximal 0.0070
	decision 0.0063	user 0.0075	unexpected 0.0055	documents 0.0076	cabin-level 0.0069	sentences 0.0060
	classifier 0.0062	cation 0.0072	knowledge 0.0054	labeled 0.0075	mining 0.0068	product 0.0058
	class 0.0061	distributed 0.0071	changes 0.0051	customer 0.0074	voting 0.0068	receiver 0.0053
	incremental 0.0056	decision 0.0070	MOLFEA 0.0050	approach 0.0073	study 0.0066	positive 0.0049
	Bayes 0.0053	models 0.0067	human 0.0049	learning 0.0067	tree 0.0055	internal 0.0044
			(2002 THEME5)			
THEME4		manufacturing 0.0122	(rule 0.0250)	clustering 0.0070	copies 0.0063	inference 0.0071
		problems 0.0099	(optimal 0.0094)	retrieval 0.0065	stream 0.0059	failing 0.0068
		demographic 0.0094	(clustering 0.0078)	SyMP 0.0058	MLC 0.0056	Learning 0.0061
		customers 0.0081	(ROCCH 0.0075)	estimators 0.0057	generation 0.0052	TiVo 0.0059
		training 0.0079	(smoothing 0.0073)	very 0.0054	Data 0.0051	itemsets 0.0059
		similarity 0.0074	(association 0.0068)	complexity 0.0053	2003 0.0047	shopping 0.0058
		yield 0.0070	(sets 0.0066)	different 0.0049	image 0.0047	Pattern 0.0057
		SVMs 0.0065	(causality 0.0065)	behavior 0.0049	frequent 0.0044	image 0.0056
		cost 0.0062	(corpus 0.0064)	each 0.0049	part 0.0041	dense 0.0055
		measures 0.0058	(term 0.0064)	small 0.0047	important 0.0041	constraints 0.0052

Table 8.5: Theme spans extracted from KDD Abstract Data

theme, although its absolute strength is decreasing for most of the time. Aids from China becomes the strongest theme in the last time period. Comparing plots of CNN and Xinhua, we see the life cycles of the correlated themes in the two data sets exhibit comparable patterns.

8.4.3 Experiments on KDD Abstracts

The publication year naturally suggests a non-overlapping partition of the KDD abstract data. We thus treat all the abstracts published in one year as one time interval. The theme spans extracted from each year using the mixture model with $\lambda_B = 0.9$ are shown in Table 8.5. The number of themes slightly differs from year to year because we apply a threshold to select only the most salient themes as described in Section 8.3.1. Similar to what we have seen on the news data, the themes here are also mostly meaningful in the context of KDD publications. The three themes in the year of 1999 are about association rule mining, clustering, and classification respectively, which are all traditional data mining topics, compared with the new topics, such as spatial data mining (theme 1) and mining in gene and microarray (theme 2), extracted in the year of 2004.

A theme evolution graph extracted using an evolution distance threshold of $\xi = 12.5$ is shown in Figure 8.9, where we see several interesting theme threads.

Thread *a* starts with theme 3 in 1999, about classification. It first evolves into theme 1 in 2001 (typical

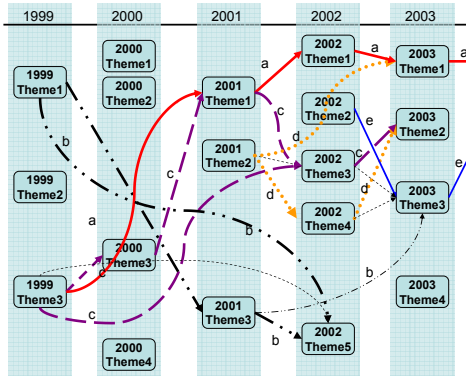


Figure 8.9: Theme evolution graph in KDD Abstract Data

classification techniques such as SVM), and then evolves into web classification in 2002. The next theme span on this thread is about clustering and random variables, which has an impact on theme 1 in 2004. Another evolution thread (b) starts with association rule mining in 1999, and transits into frequent item set in 2001. Both themes show strong evolutionary relations to theme 5 in 2002, and the frequent item set theme shows some weak evolutionary influence on theme 3 in 2003 (mining with gene expressions).

Another interesting group of edges are tagged with symbol c. Starting with classification in 1999, it transits into theme 3 in 2000 (statistical analysis and decision making). The path further connects theme 1 in 2001, and then makes an interesting transition into text classification in 2002. Theme 3 in 1999 itself also has a strong evolutionary relation to text classification. This path ends at theme 2 in 2003, which covers the Web and social networks. The discussion of web mining doesn't appear as an explicit theme until theme 2 in 2001, which evolves into web and social networks in 2003 through theme 1 in 2002.

Before year 2000, there was no explicit theme about data mining in biological data. There are two themes (theme 1, theme 2) which mention protein functionalities, but they fail to reappear in 2001. A strong explicit theme evolution thread of mining biological data (path e) starts with theme 2 in year 2002, which covers gene and time series data mining. This theme evolves into theme 3 in 2003 (mining gene expressions) and theme 4 in 2004 (mining gene and microarrays), respectively.

Theme 1	Theme 2	Theme 3	Theme 4	Theme 5	Theme 6	Theme 7
marketing 0.0087	gene 0.0173	clustering 0.0082	set 0.0076	tree 0.0107	rules 0.0142	distance 0.0150
customer 0.0086	genes 0.0104	Web 0.0070	series 0.0076	decision 0.0094	association 0.0064	objects 0.0094
models 0.0079	expression 0.0096	selection 0.0064	manufacturing 0.0066	classification 0.0086	support 0.0063	reduction 0.0071
customers 0.0076	probability 0.0081	user 0.0056	time 0.0065	itemsets 0.0061	rule 0.0060	clustering 0.0061
business 0.0048	time 0.0063	text 0.0050	clustering 0.0065	Bayes 0.0057	framework 0.0050	similarity 0.0052
no-show 0.0042	coherent 0.0058	distance 0.0048	test 0.0056	estimates 0.0052	outliers 0.0048	Euclidean 0.0050
Web 0.0041	microarray 0.0038	pages 0.0040	patterns 0.0050	probability 0.0052	useful 0.0040	dimensionality 0.0043

Table 8.6: Trans-collection themes extracted from KDD Abstract Data

As in the news data, we also analyzed the life cycles of trans-collection themes in KDD Abstracts. Seven

dominating trans-collection themes are shown in Table 8.6 and the interesting patterns of life cycles are presented in Figure 8.10. Some new topics, such as spatial-temporal data mining, have not shown up as trans-collection themes, because when we consider the whole stream, they are not among the dominating topics. From Figure 8.10, we see that the normalized strength of theme 1 decreases monotonically from 1999. This theme suggests a traditional application topic of data mining which corresponds to marketing, business and customer analysis. Another theme showing a decaying pattern is association rule mining, which keeps decreasing after its peak in 2000. In the year 1999, there is very little work on mining web information. This topic keeps growing in the following three years, and drops a bit after its acme in the year 2002. Mining from genes and biology data, as highlighted, keeps increasing over the 6 years from a very low level to one of the strongest themes. Theme 4, which covers time series and other applications of clustering, shows a irregular pattern before 2002 but remains stable after that.

There are also themes, such as classification (theme 5) and clustering (theme 7, mostly theoretical aspect, especially dimension reduction), which are somehow stable. Indeed, the classification theme appears to be among the strongest themes over the whole time line. Considering that several themes (theme 3, theme 4, and theme 7) all cover clustering, we may also infer that clustering is another major dominating theme in KDD publications.

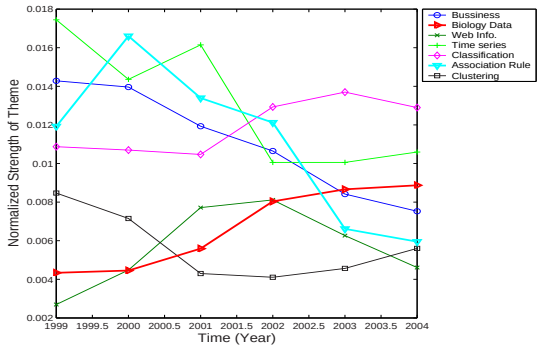


Figure 8.10: Life cycle patterns of normalized strength of trans-collection themes in KDD data

8.4.4 Parameter Tuning

In our methods for ETP discovery, there are a few parameters that are meant to provide the necessary flexibility for a user to control the pattern analysis results. We now discuss them in some detail.

In the mixture model for theme extraction, λ_B controls the strength of our background model, which is used to "absorb" non-informative words from the themes. In general, λ_B should be set to reflect a user's knowledge about the noise (i.e., the non-informative common words) in the text stream; the more noise we

believe our data set has, the larger λ_B should be. Our experiments have shown that, in ordinary English text, the value of λ_B can be set to a value between 0.9 and 0.95. Within this range, the setting of λ_B does not affect the extracted themes significantly, but it does affect the top words with the highest probabilities; a smaller λ_B tend to cause non-informative common words to show up in the top word list. Parameter k represents the number of subtopics in a subcollection that that one believes prior to theme extraction. In our experiments, we determine the number of themes by enumerating multiple possible values of k and drop the themes with significant low value of $\frac{1}{|\mathcal{D}_i|}\sum_{d\in\mathcal{D}_i}\pi_{d,j}$.

Another parameter is the evolution distance threshold ξ. This parameter has a "zooming" effect for the discovered theme evolution graph. A tighter (smaller) ξ would only allow us to see the strongest evolutionary transitions, whereas a looser (larger) ξ would allow us to examine some weaker evolutionary transitions as well.

Yet another parameter is the size of sliding window W, which is used in analyzing theme life cycles. W controls the amount of supporting documents when computing the strength of theme θ at time t. When W is set smaller, the strength curve of theme θ over time would be more precise and sensitive. However, the sharp variations make it difficult to see the overall trend of themes. When W is set larger, the strength curve would be smoother. However, some local variation patterns, which may be useful sometimes, would also be smoothed out. Regarding the "report delay" problem in the news domain, a reasonable value for W appears to be 7-15 days (3-7 days at each side of time t).

8.5 Related Work

While temporal text mining has not been well studied, there are several lines of research related to our work. Temporal data mining on numerical data has been well studied [19, 70, 160]. However, these techniques cannot be used for discovering theme evolution patterns from a text stream. Novelty Detection and Event Tracking [13, 144, 99] aims to detect the emerging of a new topic and identify boundaries of existing events. Morinaga and Yamanishi's work in 2004 tracks the dynamics of topic trends in real time text stream [124]. They assume the dynamics of each topic bears a Gaussian distribution and use a finite mixture model to learn the distribution online. The major differences between our work and their work are that (1) they do not consider relations between topics; (2) we aim to mine ETP in large static text collections while they focus on detecting topic trends online. Trend Detection [74, 150] detects emerging trends of topics from text. However, those works either don't represent topics with themes, or don't use unsupervised methods to discover themes and trends.

A related work to theme life cycle analysis is [137], where Perkio and others used a Multinomial PCA model to extract themes from a text collection and a summation of a hidden theme-document weight, which is similar to $\pi_{d,j}$ in Section 8.3.1, over all documents in a time period to represent the strength of theme j in this time period. The major difference between our work and theirs is that we model the theme transitions with an HMM which explicitly segments the whole text stream into corresponding themes, and use the size of segments of each theme to measure the theme strength.

Text clustering is another well studied problem relevant to our work. Specifically, the aspect models studied in [61, 184, 10] are related to the mixture theme model we use to extract themes. However, these works do not consider temporal structures in text. Nallapati and others studied how to discover sub-clusters in a news event and structure them by their dependency, which could also generate a graph structure [126]. A major difference between our work and theirs is that they perform document level clustering, while we perform theme level word clustering. Another difference is that they do not consider the variations of subtopics in different time periods while we analyze life cycles of themes.

Since a theme evolution graph and theme life cycle can serve as a good summary of a collection, our work is also partially related to document summarization (e.g., [77, 2]). Allan and others presented a news summarization method based on ranking and selecting sentences obeying temporal order [2]. However, summarization intends to retain the explicit information in text in order to maintain fidelity, while we aim at extracting non-obvious implicit themes and their evolutionary patterns.

8.6 Summary

Text streams often contain latent temporal theme structures which reflect how different themes influence each other and evolve over time. Discovering such evolutionary theme patterns can not only reveal the hidden topic structures, but also facilitate navigation and digest of information based on meaningful thematic threads. In this chapter, we propose general probabilistic approaches to discover evolutionary theme patterns from text streams in a completely unsupervised way. To discover the evolutionary theme graph, our method would first generate word clusters (i.e., themes) for each time period and then use the Kullback-Leibler divergence measure to discover coherent themes over time. Such an evolution graph can reveal how themes change over time and how one theme in one time period has influenced other themes in later periods. We also propose a method based on hidden Markov models for analyzing the life cycle of each theme. This method would first discover the globally interesting themes and then compute the strength of a theme in each time period. This allows us to not only see the trends of strength variations of themes, but also compare the relative strengths

of different themes over time.

We evaluated our methods using two different data sets. One is a stream of 50 days' news articles about the tsunami disaster that happened recently in Asia, and the other is the abstracts of the KDD conference proceedings from 1999 to 2004. In both cases, the proposed methods can generate meaningful temporal theme structures and allow us to summarize and analyze the text data from temporal perspective. Our methods are generally applicable to any text stream data and thus have many potential applications in temporal text mining.

This chapter serves as an illustration on how to postprocess the basic contextual patterns extracted from the context modeling phase of contextual text mining, in order to extract refined contextual patterns.

Evolutionary theme patterns, including theme evolution transition, theme evolution thread, evolutionary theme graph, and theme life cycles are good examples of refined contextual patterns. Using contextual topic models such as PLSA, what we can get are basic contextual patterns, the conditional distributions like $p(w|\theta)$ and $p(\theta|d)$. In this chapter, we presented how to further analyze these basic patterns and extract the refined patterns such as the evolutionary theme graph and theme life cycle. The former is constructed by comparing the similarity between context models using KL-Divergence. The latter is constructed by figuring out the *active domain* of each context with the help of a hidden Markov model. These two ways of patten postprocessing not only apply in this particular application, but also apply to other tasks with different contexts and contextual language models involved.

Chapter 9

Postprocessing - Automatic Labeling of Topic Models

A critical problem that is common in all the text mining problems using probabilistic topic models, including what we have discussed so far in this thesis, is how to help users interpret the discovered topics based on the corresponding multinomial distribution of words. In most existing literature, it is expected that the user would infer the meaning of a topic based on the top ranked words according to $p(w|\theta)$. However, the top ranked words in such a distribution usually cannot clearly represent the semantics carried by the distribution. In this chapter, we study another general method to postprocess contextual patterns - how to annotate a context model (such as a multinomial topic model) with meaningful labels to help users interpret and make use of discovered topics.

The techniques introduced in this chapter are not only applicable to multinomial topic models, but also applicable to other context models, as long as it is represented by a multinomial distribution of words. In this chapter, we use a topic as an example of the context, thus the meaningful labels are generated for topic models. One may imagine that when there are other types of context involved, especially implicit contexts, we may use the same technique to generate meaningful labels for the word distributions corresponding to the context models $\mathcal{M}_c$. Labeling topic models, or context models in general, is an important and general postprocessing procedure of the contextual language models and contextual patterns in contextual text mining. Through the meaningful labels, a user can quickly understand the meanings of a topic, a context, as well as the semantic variation of a topic over different contexts (see [112]).

9.1 Overview

Statistical topic modeling has attracted much attention recently in machine learning and text mining [61, 10, 184, 162, 51, 8, 111, 114, 90, 171] due to its broad applications, including extracting scientific research topics [51, 8], temporal text mining [113, 171], spatiotemporal text mining [111, 114], author-topic analysis [162, 114], opinion extraction [184, 111], and information retrieval [61, 181, 173]. Common to most of this work is the idea of using a multinomial word distribution (also called a unigram language model) to model

a topic in text.

For example, the multinomial distribution shown on the left side of Table 9.1 is a topic model extracted from a collection of abstracts of database literature. This model gives high probabilities to words such as "view", "materialized", and "warehouse," so it intuitively captures the topic "materialized view." In general, a different distribution can be regarded as representing a different topic.

Many different topic models have been proposed, which can extract interesting topics in the form of multinomial distributions automatically from text. Although the discovered topic word distributions are often intuitively meaningful, a major challenge shared by all such topic models is to accurately interpret the meaning of each topic. Indeed, it is generally very difficult for a user to understand a topic merely based on the multinomial distribution, especially when the user is not familiar with the source collection. It would be hard to answer questions such as "What is a topic model about?" and "How is one distribution different from another distribution of words?".

Without an automatic way to interpret the semantics of topics, in existing work of statistical topic modeling, people generally either select top words in the distribution as primitive labels [61, 10, 51, 8], or generate more meaningful labels manually in a subjective manner [113, 111, 114, 171]. However, neither of these options is satisfactory. Consider the following topic extracted from a collection of database literature:

Topic Model		**Variant Labels**
views	0.10	**Top Terms**: views, view, materialized,
view	0.10	maintenance, warehouse, tables
materialized	0.05	**Human**: materialized view, data warehouse
maintenance	0.05	
warehouse	0.03	**Single Term**: view, maintenance;
tables	0.02	**Phrase**: data warehouse, view maintenance
summary	0.02	**Sentence**: Materialized view selection and
updates	0.02	maintenance using multi-query optimization

Table 9.1: Variant possible labels for a topic model

It is difficult for someone not familiar with the database domain to infer the meaning of the topic model on the left just from the top terms. Similar examples can be found in scientific topics, where extracting top terms is not very useful to interpret the coherent meaning of a topic. For example, a topic labeled with "insulin glucose mice diabetes hormone"[1] may be a good topic in medical science, but makes little sense to common audience.

Manual labeling also has its own problems. Although manually generated labels are usually more understandable and better capture the semantics of a topic (see Table 9.1), it requires a lot of human effort to generate such labels. A more serious problem with manual labeling is that the labels generated are usu-

[1] www.cs.cmu.edu/∼lemur/science/topics.html, Topic 26

ally *subjective* and can easily be biased towards the user's personal opinions. Moreover, relying on human labeling also makes it hard to apply such topic models to online tasks such as summarizing search results.

Thus it is highly desirable to automatically generate meaningful labels for a topic word distribution so as to facilitate interpretations of topics. However, to the best of our knowledge, no existing method has been proposed to automatically generate labels for a topic model or a multinomial distribution of words, other than using a few top words in the distribution to label a topic. In this chapter, we study this fundamental problem which most statistical topic models suffer from and propose probabilistic methods to automatically label a topic.

What makes a good label for a topic? Presumably, a good label should be understandable to the user, could capture the meaning of the topic, and distinguish a topic from other topics. In general, there are many possible choices of linguistic components as topic labels, such as single terms, phrases, or sentences. However, as we could learn from Table 9.1, single terms are usually too general and it may not be easy for a user to interpret the combined meaning of the terms. A sentence, on the other hand, may be too specific, thus it could not accurately capture the *general* meaning of a topic. In between these two extremes, a phrase is coherent and concise enough for a user to understand, while at the same time, it is also broad enough to capture the overall meaning of a topic. Indeed, when labeling topic models manually, most people prefer phrases [113, 111, 114, 171]. In this chapter, we propose a probabilistic approach to automatically labeling topic models with meaningful phrases.

Intuitively, in order to choose a label that captures the meaning of a topic, we must be able to measure the "semantic distance" between a phrase and a topic model, which is challenging. We solve this problem by representing the semantics of a candidate label with a word distribution and casting this labeling problem as an optimization problem involving minimizing the Kullback-Leibler divergence between the topic word distribution and a candidate label word distribution, which can be further shown to be maximizing mutual information between a label and a topic model.

The proposed methods are evaluated using two text data sets with different genres (i.e., literature and news). The results of experiments with user study show that the proposed labeling methods are quite effective and can automatically generate labels that are meaningful and useful for interpreting the topic models.

Our methods are general and can be applied to labeling a topic learned through all kinds of topic models such as PLSA, LDA, and their variations. Indeed, it can be applied as a post-processing step to any topic model, as long as a topic is represented with a multinomial distribution over words. Moreover, the use of our method is not limited to labeling topic models; our method can also be used in any text management

tasks where a multinomial distribution over words can be estimated, such as labeling document clusters and summarizing text. By switching the context where candidate labels are extracted and where the semantic distance between a label and a topic is measured, we can use our method to generate labels that can capture the content variation of the topics over different contexts, allowing us to interpret topic models from different views. Thus our labeling methods also provide an alterative way of solving a major task of contextual text mining [114].

The rest of the chapter is organized as follows. In Section 9.2, we formally define the problem of labeling multinomial topic models. In Section 9.3, we propose our probabilistic approaches to generating meaningful phrases as topic labels. The variation of this general method is discussed in Section 9.4, followed by empirical evaluation in Section 9.5, discussion of related work in Section 9.6, and our conclusions in Section 9.7.

9.2 Problem Formulation

Given a set of latent topics extracted from a text collection in the form of multinomial distributions, our goal is, informally, to generate understandable semantic labels for each topic. We now formally define the problem of topic model labeling. We begin with a series of useful definitions.

Definition 9.1 (Topic Label) A *topic label*, or a "*label*", l, for a topic model θ, is a sequence of words which is semantically meaningful and covers the latent meaning of θ.

Words, phrases, and sentences are all valid labels under this definition. In this chapter, however, we only use phrases as topic labels.

For the example above, a reasonable label may be "supporting vector machine."

Definition 9.2 (Relevance Score) The *relevance score* of a label to a topic model, $s(l, \theta)$, measures the semantic similarity between the label and the topic model. Given that l_1 and l_2 are both meaningful candidate labels, l_1 is a better label for θ than l_2 if $s(l_1, \theta) > s(l_2, \theta)$.

With these definitions, the problem of **Topic Model Labeling** can be defined as follows:

Given a topic model θ extracted from a text collection, the problem of *single topic model labeling* is to (1) identify a set of candidate labels $L = \{l_1, ..., l_m\}$, and (2) design a relevance scoring function $s(l_i, \theta)$. With L and s, we can then select a subset of n labels with the highest relevance scores $L_\theta = \{l_{\theta,1}, ..., l_{\theta,n}\}$ for θ.

This definition can be generalized to label multiple topics. Let $\Theta = \{\theta_1, ..., \theta_k\}$ be a set of k topic models, and $L = \{l_1, ..., l_m\}$ be a set of candidate topic labels. The problem of *multiple topic model labeling* is to select a subset of n_i labels, $L_i = \{l_{i,1}, ..., l_{i,n_i}\}$, for each topic model θ_i. In most text mining tasks, we would

need to label multiple topics.

In some scenarios, we have a set of well accepted candidate labels (e.g., the Gene Ontology entries for biological topics). However, in most cases, we do not have such a candidate set. More generally, we assume that the set of candidate labels can be extracted from a reference text collection, which is related to the meaning of the topic models. For example, if the topics to be labeled are research themes in data mining, the reasonable labels could be extracted from the KDD conference proceedings. In most text mining tasks, it would be natural to use the text collection to be mined as our reference text collection to extract candidate labels.

Therefore, a natural work flow for solving the topic labeling problem would be (1) extracting a set of candidate labels from a reference collection; (2) finding a good relevance scoring function; (3) using the score to rank candidate labels w.r.t. each topic model; and (4) select top ranked ones to label the corresponding topic.

However, many challenges need to be solved in order to generate good topic labels automatically. As discussed in Section 9.1, a good set of labels for a topic should be (1) understandable, (2) semantically relevant, (3) covering the whole topic well, and (4) discriminative across topics. Without prior domain knowledge, extracting understandable candidate labels is non-trivial. Since a topic model and a label have different representations, it is also difficult to compare their semantics. As a result, there is no existing method to measure the semantic relevance between a topic model and a label. Even with a good measure for semantic relevance, it is still unclear how we can ensure that the label would fully cover the meaning of a topic and also capture the difference between different topic models.

In the next section, we propose a probabilistic approach to generating labels for topic models automatically.

9.3 Probabilistic Topic Labeling

To generate labels that are ***understandable***, semantically ***relevant***, ***discriminative*** across topics, and of ***high coverage*** of each topic, we first extract a set of understandable candidate labels in a preprocessing step, then design a relevance scoring function to measure the semantic similarity between a label and a topic, and finally propose label selection methods to address the inter-topic discrimination and intra-topic coverage problems.

9.3.1 Candidate Label Generation

As discussed in Section 9.1, compared with single terms and sentences, phrases appear to be more appropriate for labeling a topic. Therefore, given a reference collection $\mathcal{D}$, the first task is to generate meaningful phrases as candidate labels. Phrase generation has been addressed in existing work [31, 180, 102, 21]. In general, there are two basic approaches:

Chunking/Shallow Parsing: Chunking (Shallow Parsing) is a common technique in natural language processing, which aims at identifying short phrases, or "chunks" in text. A chunker often operates on text with part of speech tags, and uses the tags to make decisions of chunking according to some grammar, or through learning from labeled training sets. In our work, we extract the chunks/phrases frequently appearing in the collection as candidate labels.

The advantage of using an NLP chunker is that the phrases generated are grammatical and meaningful. However, the accuracy of chunking usually depends heavily on the domain of the text collection. For example, if the model is trained with news articles, it may not be effective on scientific literature. Even in scientific literature, processing biology literature is much different from processing computer science publications.

Ngram Testing: Another type of method is to extract meaningful phrases from word ngrams based on statistical tests. The basic idea is that if the words in an ngram tend to co-occur with each other, the ngram is more likely to be an n-word phrase.

There are many methods for testing whether an ngram is a meaningful collocation/phrase [31, 180, 4, 102]. Some methods rely on statistical measures such as mutual information [31]. Others rely on hypothesis testing techniques. The null hypothesis usually assumes that "the words in an ngram are independent", and different test statistics have been proposed to test the significance of violating the null hypothesis. Two famous hypothesis testing methods showing good performance on phrase extraction are χ^2 Test and Student's T-Test [102].

The advantage of such an ngram testing approach is that it does not require training data, and is applicable to text collection of any ad hoc domains/topics. The disadvantage is that the top ranked ngrams sometimes are not linguistically meaningful, and it usually only works well for bigrams.

In our experiments, we compare both approaches to extract the set of candidate labels.

9.3.2 Semantic Relevance Scoring

We propose two relevance scoring functions to rank labels by their semantical similarity to a topic model.

The Zero-Order Relevance

The semantics of a latent topic θ is fully captured by the corresponding multinomial distribution. Intuitively any reasonable measure of the semantic relevance of a label to a topic should compare the label with this distribution in some way.

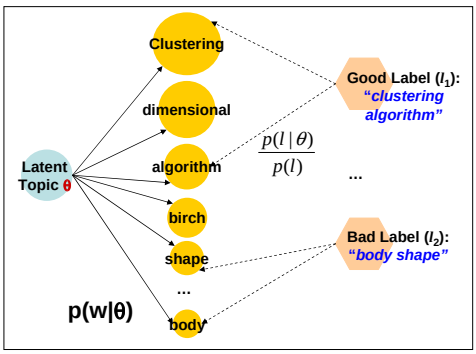


Figure 9.1: Illustration of zero-order relevance
A larger circle means a higher probability.

One possibility is to define the semantic relevance score of a candidate phrase $l = u_0 u_1 ... u_m$ (u_i is a word) as

$$Score = \log \frac{p(l|\theta)}{p(l)} = \sum_{0 \leq i \leq m} \log \frac{p(u_i|\theta)}{p(u_i)}$$

where the independence of $u_i's$ is assumed. The basic idea of this approach is illustrated in Figure 9.1. Basically, a phrase containing more "important" (high $p(w|\theta)$) words in the topic distribution is assumed to be a good label. $p(u_i)$ is to correct the bias toward favoring short phrases and can be estimated using some background collection B, or simply set to uniform. With this method, we essentially score a candidate phrase based on the likelihood that the phrase is "generated" using the topic model θ as opposed to some background word distribution.

We say that this method captures the "zero-order relevance" since no context information from the reference collection is considered. Although this method is simple and intuitively easy to understand, the semantic information of the label is ignored and the information carried by the entire topic distribution is not fully utilized. A highly ranked label may happen to consist of many high probability words but have quite different meaning from the topic. A topic in computer science containing "tree" and "apple" may not be about "apple tree". We now propose another method based on deeper analysis of semantics.

The First-order Relevance

The semantics of a topic model should be interpreted in a context. For example, a topic about "rule", "association", "correlated", "frequency" is difficult to be labeled without a context of data mining. To "decode" the meaning of the topic conveyed by a multinomial distribution, a suitable context should be considered. In such a context, terms with higher probabilities in the distribution are more likely to appear when the topic θ is covered in a document. A natural context to interpret a topic is the original collection from which the topic model is extracted.

As discussed in Section 9.2, one challenge in topic labeling is the mismatch of the representation of a topic model and a label. Our idea is thus to represent a candidate label also with a multinomial distribution of words, which we can then use to compare with the topic model distribution to decide whether the label and the topic have the same meaning. Ideally, let us assume that there is also a multinomial distribution $\{p(w|l)\}$ decided by label l. We can measure the closeness of $\{p(w|l)\}$ and $\{p(w|\theta)\}$ using the Kullback-Leibler(KL) divergence $D(\theta||l)$. Intuitively, this KL divergence can capture how good l is as a label for θ. If l is a perfect label for θ, these two distributions should perfectly match each other, thus the divergence would be zero. The basic idea of this method is illustrated in Figure 9.2.

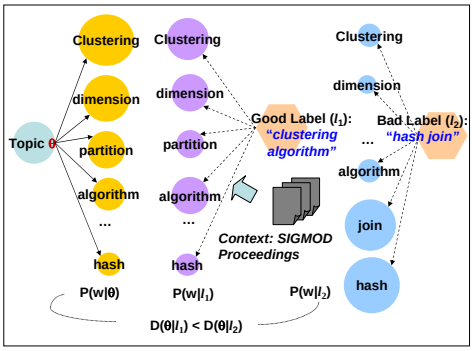


Figure 9.2: Illustration of first order relevance
A larger circle means a higher probability.

Unfortunately, there is no clue about this unknown distribution $\{p(w|l)\}$. To use this relevance score, we thus would need to approximate $\{p(w|l)\}$. One way to approximate $\{p(w|l)\}$ is to include a context collection $\mathcal{D}$, and estimate a distribution $\{p(w|l,\mathcal{D})\}$ to substitute $\{p(w|l)\}$. This approximation is reasonable: Consider the scenario when a person is unfamiliar with the meaning of a phrase, he/she would look at the context of the phrase first, and then decide whether the phrase is good to label a topic. For example, as in Figure 9.2, to label a database research topic, a reasonable context could be the SIGMOD conference proceedings. "Clustering algorithm" is a much better label for θ than "hash join" is, because the multinomial distribution

estimated based on the context of "clustering algorithm" better matches the topic distribution θ than that based on the context of "hash join." We refer to the reference collection $\mathcal{D}$ as the ***context*** of topic model labeling.

Relevance Scoring Function

Formally, the relevance scoring function of label l w.r.t. topic model θ is defined as the negative KL divergence of $\{p(w|\theta)\}$ and $\{p(w|l)\}$. With the introduction of the context $\mathcal{D}$, this scoring function can be rewritten as follows:

$$\begin{aligned}
Score(l,\theta) &= -D(\theta||l) = -\sum_w p(w|\theta)\log\frac{p(w|\theta)}{p(w|l)} \\
&= -\sum_w p(w|\theta)\log\frac{p(w|\mathcal{D})}{p(w|l,\mathcal{D})} - \sum_w p(w|\theta)\log\frac{p(w|\theta)}{p(w|\mathcal{D})} \\
&\quad -\sum_w p(w|\theta)\log\frac{p(w|l,\mathcal{D})}{p(w|l)} \\
&= \sum_w p(w|\theta)\log\frac{p(w,l|\mathcal{D})}{p(w|\mathcal{D})p(l|\mathcal{D})} - D(\theta||\mathcal{D}) \\
&\quad -\sum_w p(w|\theta)\log\frac{p(w|l,\mathcal{D})}{p(w|l)} \\
&= \sum_w p(w|\theta)PMI(w,l|\mathcal{D}) - D(\theta||\mathcal{D}) + Bias(l,\mathcal{D})
\end{aligned}$$

From this rewriting, we see that the scoring function can be decomposed into three components. The second component is the KL divergence between the topic and the labeling context. Intuitively, if we use humanity literature as the context to label a data mining topic, the relevance score will be lower since it is not as trustworthy as computer science literature. However, this divergence is identical for all candidate labels, thus can be ignored in ranking labels. The third component can be viewed as a bias of using context $\mathcal{D}$ to infer the semantic relevance of l and θ. Consider the scenario that l is a good label for θ according to some prior knowledge, such as a domain ontology, but does not appear in $\mathcal{D}$. In this case, $\mathcal{D}$ is biased to be used to infer the semantics of l w.r.t. θ. Practically, $Bias(l,\mathcal{D})$ can be utilized to incorporate priors of candidate labels. When both the topic models and the candidate labels are generated from the collection $\mathcal{D}$, we simply assume that there is no bias.

Interestingly, the first component can be written as the expectation of pointwise mutual information between l and the terms in the topic model given the context ($E_\theta(PMI(w,l|\mathcal{D}))$). Without any prior knowledge on the label-context bias, we rank the candidate labels with $E_\theta(PMI(w,l|\mathcal{D}))$, where

$PMI(w, l|\mathcal{D})$ can all be pre-computed independently of the topic models to be labeled.

Note that $PMI(w, l|\mathcal{D})$ is undefined if $p(w, l|\mathcal{D}) = 0$. One simple strategy is to ignore such w in the summation. A more reasonable way is to smooth $p(w, l|\mathcal{D})$ with methods like Laplace smoothing.

This relevance function is called the first-order relevance of a label to a topic.

Intuitive Interpretation

Ranking candidate labels based on $E_\theta(PMI(w, l|\mathcal{D}))$ is technically well motivated. However, is this a reasonable formalization in reality? What does this ranking function essentially capture? In this section, we give an intuitive interpretation of this semantic relevance scoring function.

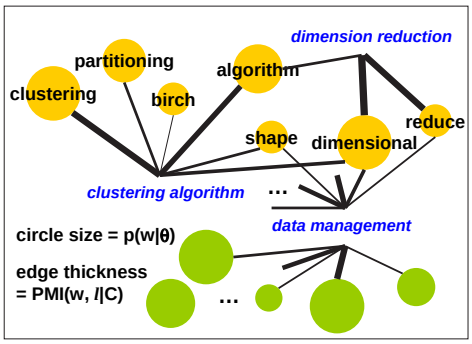


Figure 9.3: Interpretation of label selection
A label having high PMI with many high probability topic words would be favored.

As shown in Figure 9.3, we can construct a weighted graph, where each node is either a term in a topic model (weighted with $\{p(w|\theta)\}$) or a candidate label. Each edge between a label and a topical term is then weighted with the pointwise mutual information $PMI(w, l|\mathcal{D})$, which is often used to measure semantic associations [31, 102, 135]. Thus the weight of each node indicates the importance of the term to this topic, while the weight of each edge indicates how strongly the label and the term are semantically associated.

The scoring function $E_\theta(PMI(w, l|\mathcal{D}))$ would rank a label node higher, if it generally has *stronger semantic relation* (thicker edge) to those *important topical words* (larger circles). Intuitively, the labels selected in this way are meant to cover the entire topic model well.

9.3.3 High Coverage Labels

The third criterion of a good label is the high intra-topic coverage. We expect a label to cover as much semantic information of a topic as possible. Indeed, if we only extract one label for each topic, the semantic relevance function already guarantees that the the label covers maximum semantic information of θ. However,

one label usually only partially covers a topic. When selecting multiple labels, we naturally expect the new labels to cover different aspects of the topic, not the information covered by the labels already selected.

Intuitively, in Figure 9.3, let us assume that "clustering algorithm" is already selected to label the upper topic. However, there are still important topical nodes (e.g., "dimensional", "reduce") weakly covered, or not covered by the label. We thus expect the second label to cover this missing information as much as possible, thus we prefer "dimension reduction", rather than labels like "clustering technique".

To implement this intuition, we propose to select labels with the Maximal Marginal Relevance (MMR) [16] criterion. MMR is commonly used in information retrieval tasks, where both high relevance and low redundancy of retrieval results are desired.

Specifically, we select labels one by one, by maximizing the MMR criterion when selecting each label:

$$\hat{l} = \arg\max_{l \in L-S} [\lambda Score(l, \theta) - (1-\lambda) \max_{l' \in S} Sim(l', l)]$$

where S is the set of labels already selected, $Sim(l', l) = -D(l'||l) = -\sum_w p(w|l') \log \frac{p(w|l')}{p(w|l)}$, and λ is a parameter to be empirically set.

9.3.4 Discriminative Labels

The previous criteria all only consider the labeling of a single topic. When a set of topics are presented, achieving inter-topic discrimination would be another criterion to consider. A label with high relevance scores to many topic models would not be very useful in this case even though it may be a good label for each individual topic. Intuitively, in Figure 9.3, "clustering algorithm" is a better label for the upper topic than "data management", since the former covers the important nodes well and exclusively.

In principle, we expect a good label to have high semantic relevance to the target topic model, and low relevance to other topic models. We thus propose the following modified scoring function:

$$Score'(l, \theta_i) = Score(l, \theta_i) - \mu Score(l, \theta_{1,\ldots,i-1,i+1,\ldots,k})$$

where $\theta_{1,\ldots,i-1,i+1,\ldots,k}$ (short as θ_{-i}) is the semantics carried by all other topics than θ_i, and μ controls the

discriminative power. $Score(l, \theta_{-i})$ can be modeled as

$$\begin{aligned} Score(l, \theta_{-i}) &= -D(\theta_{-i}||l) \\ &\stackrel{\text{rank}}{=} E_{\theta_{-i}}(PMI(w, l|\mathcal{D})) \\ &\approx \frac{1}{k-1} \sum_{j=1,..,i-1,i+1,..,k} \sum_{w} p(w|\theta_j)(PMI(w, l|\mathcal{D})) \\ &= \frac{1}{k-1} \sum_{j=1..k} E_{\theta_j}(PMI(w, l|\mathcal{D})) \\ &\quad -\frac{1}{k-1} E_{\theta_i}(PMI(w, l|\mathcal{D})) \end{aligned}$$

which leads to

$$Score'(l, \theta_i) \approx (1 + \frac{\mu}{k-1}) E_{\theta_i}(PMI(w, l|\mathcal{D})) - \frac{\mu}{k-1} \sum_{j=1..k} E_{\theta_j}(PMI(w, l|\mathcal{D}))$$

We use $Score'(l, \theta)$ to rank the labels, which achieves the needed discrimination across topic models.

With the methods proposed in this section, we are able to generate labels for multinomial topic models, which are understandable, semantically relevant, discriminative across topics, and of high coverage inside topics.

Although it is motivated to label multinomial topic models, the use of our approach is not limited to this. Variations in using the labeling approach could lead to different interesting applications. In the following section, we present two possible applications of topic model labeling.

9.4 Variations of Topic Labeling

In the previous section, we proposed the probabilistic framework and methods to label topic models, in which we assume that there is a multinomial representation for each topic model, and a context collection to generate candidate labels and measure the semantic relevance of a candidate label to a topic. In this section, we relax the assumption and introduce some variations of topic labeling, which can lead to many interesting text mining applications.

9.4.1 Labeling Document Clusters

The topic labeling framework, which is proposed to label topic models, essentially consists of a multinomial word distribution, a set of candidate labels, and a context collection. Thus it could be applied to any text mining problems, in which a multinomial distribution of word is involved.

In some tasks, such as topic modeling and many information retrieval tasks [35, 181], a multinomial word distribution is explicit. In other tasks, however, a multinomial word distribution may not be directly available; in such tasks, we can also apply our method by extracting a multinomial distribution. For example, given a group of documents $G = \{d_1, ..., d_m\}$, a multinomial word distribution can be easily constructed using the maximum likelihood estimation:

$$p_G(w) = \frac{\sum_{d \in G} c(w, d)}{\sum_{d' \in G} \sum_{w'} c(w', d')}$$

The proposed multinomial topic model labeling methods can be easily applied to generating labels for $\{p_G(w)\}$. Such labels can thus be used to interpret the original group of documents. This is extremely valuable as many text management tasks involve a group/groups of documents, whose latent semantics is difficult to present. For example, document clustering partitions a collection of documents into groups, where a good label for each group may help the user understand why these documents are grouped together.

Labeling a cluster of documents is also valuable for many other tasks, such as search result summarization and model-based feedback [181]. In fact, the topic labeling method can be applied to any mining problems where a multinomial distribution of words can be estimated, such as term clustering, annotation of frequent patterns in text.

9.4.2 Context Sensitive Labeling

Another possible variation is to switch the context collection, i.e., label a topic model extracted from one collection with another collection as the context. Although we normally would like to label a topic using the collection from which the topic is extracted as the context, it may be interesting sometimes to label/interpret a topic model in different contexts. Such cross-context interpretation can help us understand the variations of a topic and the connections between different contexts. For example, interpreting a topic discovered from one research area (e.g., database) in the context of another related research area (e.g., information retrieval) may reveal interesting connections between the two areas (e.g., an interdisciplinary research theme). Since our method can work on any context, we can easily use it to achieve such cross-context interpretation of topics, and contextual text mining in general.

Indeed, a major task of contextual text mining is to extract topics and compare their variations in different contexts (e.g., time [113], location [111], authorship [162, 114], etc). In the existing work, this is done by designing a specific statistical topic model with contextual structure, and fitting the data directly with the model. Topic labeling provides an alternative way to track the context-sensitive semantics of a

general topic. By using different context collections, the semantics of candidate labels and topic models are biased towards the context. The labeling algorithm thus can generate different labels for the same topic, from the view in different contexts. Such technique can be applied to many contextual text mining tasks, such as temporal, spatiotemporal text mining, and author-topic analysis.

In Section 9.5, we show that variations of the general topic labeling framework are effective for different mining tasks.

9.5 Experiments and Results

In this section, we present the results of our evaluation of the effectiveness of the proposed methods for automatically labeling multinomial topic models using two data sets.

9.5.1 Experiment Setup

Data Sets: We explore two different genres of document collections: the SIGMOD conference proceedings, and the Associated Press (AP) news dataset. To construct the first dataset, we downloaded 1848 abstracts of SIGMOD proceedings between the year 1975 and 2006, from the ACM digital library[2]. The second data collection contains a set of 2246 AP news articles, downloaded from http://www.cs.princeton .edu/∼blei/lda-c/ap.tgz. We built an index for each collection and implemented the topic labeling methods proposed in Section 9.3 with the Lemur toolkit[3].

Candidate Labels: We generate two sets of candidate labels with different methods: (1) extract noun phrases chunked by an NLP Chunker[4]; (2) extract most significant 2-grams using the N-gram Statistics Package [4]. We use the T-Test to test the significance of 2-grams, and extract those with the highest T-Scores [102]. More specifically, we extract the top 1000 candidate 2-grams ranked by T-Score and top 1000 chunked noun phrases ranked by their frequencies. The ngrams with the highest T-Scores and the most frequent noun phrases are presented in Table 9.2.

SIGMOD		**AP**	
2-gram	noun phrase	2-gram	noun phrase
database systems	this paper	he said	the united states
database system	the problem	more than	the government
object oriented	a set	united states	last year
query processing	the data	new york	the country
data base	the database	last year	the nation

Table 9.2: Sample candidate labels

[2]http://www.acm.org/dl
[3]http://www.lemurproject.org/
[4]http://opennlp.sourceforge.net/

Topic Models: From each dataset, we extract a number of topics using two representative statistical topic models, the PLSA [61] and LDA [10]. A background component model is added into PLSA to absorb the non-informative words, as suggested in [184] and [113]; this will make the topic models more distinguishable and readable. We do not use such a background model, or prune stopwords for LDA, in order to test the robustness of our topic labeling methods. We extracted 30 and 50 major topics from the SIGMOD and AP dataset, respectively. A subset of example topics is shown in Table 9.3, where we list the words of the highest probabilities for each topic in the bottom row. We can see that for some topics, especially those from news articles (AP), it is hard to tell the latent meaning merely from the top words.

	SIGMOD				AP			
Auto Label	**clustering algorithm**	**r tree**	**data streams**	**concurrency control**	**air force**	**court appeals**	**dollar rates**	**iran contra**
Man. Label	**clustering algorithms**	**indexing methods**	**Stream data management**	**transaction management**	**air plane crash**	**death sentence**	**international stock trading**	**iran contra trial**
θ	clustering	tree	stream	transaction	plane	court	dollar	north
	clusters	trees	streams	concurrency	air	judge	1	case
	video	spatial	continuous	transactions	flight	attorney	yen	trial
	dimensional	b	monitoring	recovery	pilot	prison	from	iran
	cluster	r	multimedia	control	crew	his	late	documents
	partitioning	disk	network	protocols	force	trial	gold	walsh
	quality	array	over	locking	accident	case	down	reagan
	birch	cache	ip	log	crash	convicted	london	charges

Table 9.3: Sample topics and system-generated labels
The second row contains the automatically generated labels. The third row presents the manually generated labels. The fourth row shows the words of highest probabilities in the topic distribution.

9.5.2 Effectiveness of Topic Labeling

We first show some sample results of our topic labeling method in Table 9.3; for comparison, we also show the human-generated labels for the same topics. It is clear that the automatically generated labels can all capture the meaning of the topic to some extent; indeed, most of them are as good as human generated labels (e.g., "clustering algorithm" and "data streams"), though some are not (e.g., "air force"). Some topics are difficult to interpret even by human (e.g., "death sentence").

To quantitatively evaluate the effectiveness of the automatic labeling methods, we ask three human assessors to compare the results generated by different methods. Specifically, for each of the most salient topics generated with PLSA (12 topics from SIGMOD and 18 topics from AP), we present to the annotators the labels generated by different methods in a random order, together with the word distribution and the most relevant documents to this topic to help a human assessor interpret the topic. A baseline method is included in comparison, which simply uses the top k terms in the word distribution as the topic labels. The other methods included in the comparison are shown in Table 9.4.

Given the labels generated by n (n = 2, 3...) systems, we ask the assessors to rank the systems according to the quality of the labels they generated. For each topic, they will assign a score of $n - k$ to a system if

System	Cand. Labels	Relevance Score
NGram-1	Ngrams	First-order
NGram-0-U	Ngrams	0-order, uniform normaliz.
NGram-0-B	Ngrams	0-order, norm. with $p(w\|B)$.
Chunk-1	NP Chunks	First-order

Table 9.4: Systems Compared in Human Evaluation
Default parameter setting: $\lambda = 0.2$, $\mu = 0.7$ for AP; $\lambda = 0.2$, $\mu = 1$ for SIGMOD

it is ranked at the k'th place. If the labels from several systems are difficult to be distinguished/ranked, we first give them an arbitrary ranking, and then equal their scores. For example, if there are three systems, one is significantly better, and the other two are hard to tell, we will assign score 2 to the first system, and 0.5 to each of the rest two systems. We then average the scores of each system over all topics.

Baseline v.s. Zero-order v.s. First-order				
Dataset	#Label	Baseline	Ngram-0-B	Ngram-1
SIGMOD	1	0.76	0.75	**1.49**
SIGMOD	5	0.36	1.15	**1.51**
AP	1	0.97	0.99	**1.02**
AP	5	0.85	0.66	**1.48**

Table 9.5: Effectiveness of topic labeling
A higher score means that the system tends to be ranked higher.

Basic results: In Table 9.5, we compare the labels generated using the baseline method (i.e., picking high probability words), 0-order relevance (ngrams, normalized with background probability $p(w|B)$), and 1st-order relevance. For each group of systems, we compare both the top 1 label, and the top 5 labels they generate. From this table, we can make several observations: (1) In all cases, the labels extracted with 1st-order relevance are most preferred by the assessors, indicating that the first-order relevance method is overall the best presumably due to the fact that it can capture the overall topic distribution through context. (2) The preference of first-order relevance labels over the baseline labels is more significant on SIGMOD than on AP. This is likely because phrases are more frequently used and more discriminative in scientific literature than in the news domain. For example, informative phrases such as "mining association rules" are quite common in database literature, whereas common phrases in news articles tend to be general terms such as "united states" and "last year." This suggests that phrases are generally good labels for scientific topics, but for other genres of text, it may be interesting to explore other candidate labels, such as short sentences. (3) The preference of the first-order relevance labels over the baseline labels is stronger when five labels are considered than when one label is considered. This may be because the preference is amplified when more labels are considered. (4) The labels of 0-order relevance seem to be comparable with those of the baseline except in one case (i.e., 5 labels on SIGMOD) when the 0-order relevance labels are strongly preferred. This again suggests that phrases are not so useful for labeling topics in the news domain, but they are more useful

for the literature domain. Overall these results show that the first-order relevance method for automatic labeling of topic models is the best among all the methods.

Noun Phrases v.s. Ngrams			
Dataset	#Label	Chunk-1	Ngram-1
SIGMOD	1	0.40	**0.60**
SIGMOD	5	0.16	**0.83**
AP	1	0.41	**0.59**
AP	5	**0.55**	0.44

Table 9.6: Ngrams vs. noun phrases as labels

Ngrams vs. noun phrases as candidate labels: To see which of the two methods for generating candidate labels (i.e., ngrams and noun phrases) is better, we compare them on both data sets in Table 9.6. Interestingly, for the SIGMOD dataset, using statistically significant ngrams as labels is much better than using noun phrases generated by the NLP Chunker, while on the AP dataset the performance of the two types of candidate labels is closer, and in some cases the noun phrases perform even better than ngrams. This may be because the models used by the NLP Chunker are trained on general domains, and not tuned for parsing scientific literature. In general, using significant ngrams appears to be more robust; moreover, this method can also be applied to any genre of texts.

Normalization in 0-order relevance: We now look into the influence of the normalization strategy on the performance of 0-order relevance. In Table 9.7, we compare 1st-order relevance with 0-order relevance when using two different normalization strategies – uniform normalization (normalization with uniform distribution) and background normalization (normalization with a background distribution). We see that background normalization, although intuitively appealing, does not really help here. The using of background normalization fails to decrease the difference between the 0-order relevance to the better method, the 1-order relevance. Indeed, it even makes the 0-order labels worse when applied on AP. The reason is because the topic models extracted with PLSA are already discriminative due to the use of a background component model (see Section 9.5.1), thus further normalization with background is not useful, and uniform normalization is actually more robust in this case.

Normalization	Dataset	NGram-0	NGram-1	Diff.
Using Uniform	SIGMOD	0.36	0.64	**0.28**
	AP	0.43	0.57	**0.14**
Using Background	SIGMOD	0.37	0.63	**0.26**
	AP	0.26	0.74	**0.48**

Table 9.7: Uniform vs. background normailzation for 0-order relevance (5 labels)

To see if background normalization is useful when the extracted topic models are not discriminative (i.e., high probability words are non-informative words), we apply the topic labeling techniques to the topic

models extracted with LDA, where neither is a background model included, nor are the stopwords pruned. The results are selectively presented in Table 9.8.

Model	Labels	Model	Labels
the, of, a, and, in, data, ...	**1st-order:** foreign key, data integration, schema matching, query rewrite, web sites, deep web	the, of a, to and, is ...	**1st-order:** iceberg cube, data cube, data cubes, two types, fact table
constraints database integration sources	**0-order(u):** data integration, data sources, real data, their data, data model, database design	data cube query system	**0-order(u):** data cube, user query, large data, data cubes, data structure, over data
content design information schema	**0-order(b):** integrity constraints, dynamic content, sql statements, foreign key, schema matching	information olap multimedia algorithm	**0-order(b):** iceberg cube, m se, data cubes, data cube, line analytical

Model	Labels
the, of, a, and to, data ...	**1st-order:** clustering algorithm, clustering structure, data bubbles, distance function, very large
clustering time clusters databases	**0-order(u):** large data, data quality, series data, data applications, high data, clustering algorithm
large performance quality algorithm	**0-order(b):** transitive closure, subsequence matching, data bubbles, clustering algorithm, pattern matching

Table 9.8: Labeling LDA topics: 1st-order relevance is most robust

In Table 9.8, we show three sample topics extracted with LDA and their corresponding labels generated using 1st-order relevance and 0-order relevance with different normalization methods. Without pruning stopwords or using a background model, we end up having many non-informative words on the top of each topic model; to better illustrate the meaning of each topic, at the bottom part of the topic word distribution, we also present some more discriminative terms. We see that the 1st-order relevance still generates good discriminative labels even though the high probability words of the original topic model are all non-informative words. This is because the 1st-order relevance captures the entire context of the topic model. In contrast, with uniform normalization, the 0-order relevance would be biased to assign high scores to non-informative phrases such as "real data" and "their data" when the top probability terms of the topic model are non-informative (e.g., "the", "their") or too general (e.g., "data", "large"). With normalization by background model $p(w)$, we can penalize a phrase with non-informative or general words, thus alleviate this problem. However, in this way, the top ranked labels tend to be too specific to cover the general meaning of the topic (e.g., "integrity constraints", "transitive closure", etc). Thus overall we see that modeling the semantic relevance with first-order relevance is most robust because the semantics is inferred based on the context of the entire distribution.

Upper bound analysis: How much room is there to further improve the topic labeling method? We can answer this question by looking into how much worse the automatically generated labels are than those generated manually. Thus we ask a human annotator to generate topic labels manually, and ask two different

assessors to compare the system-generated labels with the human-generated labels. In Table 9.9, we see that although the system-generated labels are good, the assessors still consider human-generated labels to be better. This implies that there is still much room to improve the quality of the automatically generated topic labels. Interestingly, the difference between system generated and human generated labels is less significant on the SIGMOD data than on the AP data, suggesting that literature topics may be easier to label than news topics.

System v.s. Human			
Dataset	#Label	Ngram-1	Human
SIGMOD	1	0.35	**0.65**
SIGMOD	5	0.25	**0.75**
AP	1	0.24	**0.76**
AP	5	0.21	**0.79**

Table 9.9: Comparison with human generated labels

9.5.3 Labeling Document Clusters

In Section 9.4.1, we discussed that the topic labeling method could actually be applied to any text information management tasks where a multinomial word distribution is involved. Here we look into one such application – labeling document clusters. In this experiment, we cluster the SIGMOD abstracts with the K-Medoids algorithm [69], and try to utilize the topic labeling method to label the clusters. Specifically, we estimate a multinomial word distribution for each cluster based on its member documents using the maximum likelihood estimator. The proposed topic labeling technique can then be applied on the estimated term distributions, and the top ranked phrases are used to label the original cluster.

Cluster Labels	$\|d\|$	**Cluster Medoids (Title)**
multivalued dependencies, functional dependencies	167	A complete axiomatization for functional and multivalued dependencies in database relations
two locking, concurrency control	86	Performance of B-tree concurrency control algorithms
nearest neighbor, similarity search	69	Optimal multi-step k-nearest neighbor search
approximate answering, approximate query	184	Approximate XML query answers

Table 9.10: Labeling document clusters: K-Medoids

The generated cluster labels, along with the number of documents and the title of the medoid document of each cluster are shown in Table 9.10. By comparing the cluster labels with the medoid documents, we see that the topic labeling technique is also effective to label document clusters. This experiment shows that the use of topic labeling is not limited to statistical topic modeling; it is potentially applicable to any tasks

in which such a multinomial term distribution is involved.

9.5.4 Context-Sensitive Labeling

In this section, we evaluate the effectiveness of our topic labeling method for *cross-context* labeling/interpretation of topic models. We extract 30 topics from SIGMOD proceedings, but use the phrases extracted from SIGIR abstracts, and KDD abstracts to label the topics. We simulate the scenario in which the system does not know where the topics are extracted, thus it would simply use any context collection "familiar" to the system (i.e., SIGIR or KDD collections in our experiments). The results are presented in Table 9.11.

Topic	Labels	Topic	Labels
tree	**SIGMOD Labels**	views	**SIGMOD Labels**
trees	r tree, b trees,	view	materialized views,
spatial	index structures	materialized	view maintenance, data warehouses
r	**KDD Labels**	maintenance	**KDD Labels**
b	tree algorithm, decision trees	warehouse	decision support
disk	tree construction	tables	business intelligence
dependencies	**SIGMOD Labels**	sampling	**SIGMOD Labels**
functional	multivalued dependencies,	estimation	selectivity estimation
cube	functional dependencies, iceberg cube	approximate	random sampling, approximate answers
multivalued	**SIGIR labels**	histograms	**SIGIR Labels:**
iceberg	term dependency	selectivity	distributed retrieval,
buc	independence assumption	histogram	parameter estimation, mixture models

Table 9.11: Labeling database topics with different contexts

The results are interesting. The labels generated from different contexts generally capture the *biased* meaning of the topic from the view of that context. For example, the database topic about R-tree and other index structures assigns high probability to words like "tree" and "trees"; the results show that, when interpreted in the data mining context, these high probability words may cause the topic to be labeled with "decision trees" and "tree algorithms", suggesting a different, but related interpretation of "tree." Also, our results suggest that when seeing a topic word distribution with high probability words such as "sampling" and "estimation", database researchers may interpret it as "selectivity estimation" or "approximate answers", while information retrieval researchers interpret it as about "parameter estimation" of "mixture models", which is more relevant to their background.

This experiment shows that the topic labeling technique can be exploited to infer context-sensitive semantics of a topic model through labeling a general topic with different contexts. This provides an alternative way to solve a major task in contextual text mining: extracting general topics and analyzing the variation of their meanings over contexts.

Note that this effect can be only achieved when the first-order semantic relevance is used, since the zero-order relevance is independent of a context.

9.6 Related Work

To the best of our knowledge, no existing work has formally studied the problem of automatic labeling of multinomial topic models. There has been a large body of work on statistical topic models [61, 10, 184, 162, 51, 8, 9, 111, 114, 129, 90, 171], most of which uses a multinomial word distribution to represent a topic. In some recent work, [170] generalized the representation of a topic model as a multinomial distribution over ngrams. Such topics are labeled with either top words in the distribution or manually selected phrases. The method we proposed can automatically generate meaningful phrase labels for multinomial topic models and can be applied as a post-processing step for all such topic models, to interpret the semantics of these topics models extracted from text data.

As we use phrases as candidate labels, our work is related to phrase extraction, including shallow parsing/chunking in natural language processing (e.g., [102, 55]), and N-gram phrase extraction with statistical approaches (e.g., [31, 180, 4, 21]). A better phrase extraction method could benefit topic labeling as a better preprocessing procedure.

Text summarization aims at extracting/generating sentence summaries for one/multiple documents (e.g., [142]). The summary can be as short as titles [64]. However, no existing work has been done for summarizing a multinomial distribution of words, or a statistical topic model. Since most topic models assume that a document covers multiple topics, it is also difficult to cast topic model labeling as summarizing documents. The topic labeling approach, on the other hand, provides a novel method to label a set of documents.

A major task in contextual text mining is to extract topics and compare their content variations over different contexts [184, 162, 113, 111, 114]. Our proposed topic labeling approach provides an alternative way to infer the context-sensitive semantics of topic models.

9.7 Summary

Statistical topic modeling has been well studied recently, with applications to many machine learning and text mining tasks. Despite its high impact, however, there is no existing method which could automatically generate interpretable labels capturing the semantics of a multinomial topic model. Without understandable labels, the use of topic models in real world applications is seriously limited. In this chapter, we formally study the problem of automatic labeling of multinomial topic models, and propose probabilistic approaches to label multinomial word distributions with meaningful phrases. We cast the labeling problem as an optimization problem involving minimizing Kullback-Leibler divergence between word distributions and maximizing mutual information between a label and a topic model.

Empirical experiments show that the proposed approach is effective and robust when applied on different genres of text collections to label topics generated using various statistical topic models (e.g., PLSA and LDA). The proposed topic labeling methods can be applied as a post-processing step to label any multinomial distributions in any text context. With reasonable variations, this approach can be applied to any text mining tasks where a multinomial term distribution can be estimated. This includes labeling a cluster of documents and inferring the variation of semantics of a topic over different contexts.

Labeling topic models, or context models in general, is an important postprocessing procedure for the contextual patterns extracted by the contextual language models in contextual text mining. In fact, for the proposed two relevance scoring methods, the zero-order relevance refers to ranking labels by how likely it is generated from context model of the topic; and the first-order relevance refers to ranking labels by comparing the topic model with the model of a context defined by the label.

Although we only presented the effectiveness of generating meaningful labels for topic models (i.e., every topic is a context), we expect that this general method can be applied to any other contextual language models, as long as a multinomial distribution is used to represent a context model $\mathcal{M}_c$. This postprocessing procedure is general, which bridges the gap between contextual patterns and users by interpreting the contextual patterns with understandable labels to the users. We expect to see a lot of applications of this method in real world text mining tasks.

Chapter 10

Application: Contextual Text Mining in Ad Hoc Retrieval

In the previous chapters, we have introduced a lot of applications of contextual text mining, which are executed by various instantiations of the contextual topic model. We also introduced important components in contextual text mining, including adding model priors to guide the contextual language models, regularizing the contextual language models with a graph structure of contexts, and a variety of postprocessing techniques for extracting refined contextual patterns from basic contextual patterns.

We have already shown that contextual text mining is powerful with various instantiations. In reality, the power of contextual text mining is not restricted to its own basic tasks, e.g., extracting contextual patterns, but can also be demonstrated by the effectiveness of utilizing contextual text mining to help with other important text information management tasks, such as document retrieval and web search. How can we utilize the contextual patterns extracted to facilitate text information management tasks? How to quantitatively evaluate the effectiveness of contextual text mining? How effective does contextual text mining work with large scale datasets (e.g., the web scale)? All these questions can be answered by the applications of contextual text mining in large scale text information management tasks like text retrieval and web search.

In this chapter, we study a particular application of contextual text mining in the "context" of text retrieval. The basic idea is to leverage the contextual language model with the graph-based regularization to smooth language models in ad hoc retrieval (see [116]). We apply a special case of NetPLSA model in Chapter 7 (essentially a special case of CPLSA with a regularization on either a document graph or a term graph). This study provides a quantitative evaluation of the regularized contextual language model. Some smoothing techniques induced by contextual text mining outperform the state-of-the-art smoothing methods.

10.1 Overview

Language models have attracted much attention in the information retrieval community recently due to their success in a variety of retrieval tasks [139, 35]. Fundamental to all the language models used for retrieval is the issue of smoothing, which has been shown to affect retrieval performance significantly [184]. Indeed, in the basic language modeling approaches [139, 58, 120, 182], the entire retrieval problem is essentially reduced to the problem of estimating a document language model which can be further reduced to how to smooth the document language model. In other more sophisticated use of language models, such as relevance models [84] and model-based feedback methods [181], improved smoothing of document language models is also shown to improve performance [97, 165].

Because of the importance of smoothing, it has been attracting attention ever since Ponte and Croft's pioneering work on applying language models to retrieval [139]. Since then many smoothing approaches have been proposed and tested. In early days, most smoothing methods relied on using a background language model, which is typically estimated based on the whole document collection, to smooth a document language model [139, 58, 120, 182]. Recently, corpus graph structures have been exploited to provide more accurate smoothing of document languages. The basic idea is to smooth a document language model with the documents similar to the document under consideration through either clustering or document expansion [97, 79, 80, 38, 140, 81, 165]. Such a *local* smoothing strategy can leverage document similarity structures to offer "customized" smoothing for each individual document; this is in contrast to the simple *global* smoothing strategy which smoothes all documents with the same background model.

The local smoothing strategy has been shown to be quite effective in several recent studies [97, 165, 81] and is the best smoothing strategy known so far. In virtually all these local smoothing methods, the smoothing formula eventually boils down to some form of interpolation of the original unsmoothed document language model (i.e., the maximum likelihood estimate of the unigram language model) and some "supporting" language models estimated based on documents similar to that document. Different smoothing methods differ in how they assign weights to all these different component language models. It is known that this weighting of component language models can significantly affect retrieval performance, and sometimes even a small difference in weighting can lead to visible difference in performance [97]. Unfortunately, none of the existing work offers a formal framework to optimize these weights; as a result, there is no guidance on how to further improve these existing smoothing methods or develop new (potentially better) smoothing methods. For example, it is unclear whether we can exploit other structures such as word similarity graphs to improve smoothing.

It is not hard to see that the estimation of document and query language models is a simple instantiation

of contextual text mining, where every document and every query is a context. The goal is then to estimate an accurate and robust language model $\mathcal{M}_c$ for every document and query, and then feed the estimated document/qeury models into retrieval models such as the KL-divergence retrieval model. In this way, the process of retrieval can be regarded as a postprocessing of the context models, where the key research issue is still how to estimate accurate context models (i.e., how to extract high quality contextual patterns).

In this chapter, we propose a general unified optimization framework for smoothing language models on graph structures. While not explicitly stressed in the existing work, we believe that graph structures are fundamental to smoothing; indeed, we can interpret smoothing intuitively as to make the language models of those nodes close to each other on a graph structure similar to each other, so that the surface representing all the language models would be "smooth."

Our framework unifies two major heuristic goals in smoothing within a single objective function: (1) "Fidelity": A smoothed language model should not deviate too much from the original non-smoothed language model; (2) "Smoothness": After smoothing, the nodes that are close to each other on the graph would have similar (smoothed) language models; the closer the nodes are, the more similar their language models are. It can be easily shown that the proposed smoothing framework is a special case of the NetPLSA model introduced in Chapter 7.

This framework not only provides a principled formulation of the existing smoothing heuristics but also serves as a road map for systematically exploring smoothing methods for language models. We follow this road map and derive several different instantiations of the framework, including smoothing on document graphs and smoothing on word graphs, as well as smoothing document language models and query language models. Although document graphs have been used for smoothing in the previous work, to the best of our knowledge, no previous work has studied how to smooth language models with a word graph. Moreover, existing work on smoothing has mostly attempted to smooth document language models, while we also study how to use a word graph to smooth a query language model.

These smoothing methods are evaluated using several standard TREC test collections. The results show that all the derived smoothing methods can improve over the baseline global smoothing method significantly, indicating that our framework is reasonable. The derived smoothing methods either outperform or perform similarly to the corresponding state of the art local smoothing methods.

Given the importance of smoothing in the language modeling approach to information retrieval and given the generality of our framework, we hope that the framework can open up some promising new directions for finding an optimal way of smoothing language models.

The rest of the chapter is organized as follows. In Section 10.2, we propose the general optimization

framework for smoothing language models with graph structure, and introduce a unified solution. In Section 10.3, we follow the framework and introduce four instantiations of the general framework. We discuss the properties of those instantiations in Section 10.4 and evaluate them with empirical experiments in Section 10.5. Finally, we discuss related work and conclude in Section 10.6 and 10.7, respectively.

10.2 Smoothing Language Models with Graph Structure

To motivate our framework, we start with a brief discussion of the intuitions behind the major smoothing heuristics.

10.2.1 Intuitions

Given a non-smoothed document language model $P_{ML}(w|d)$ (i.e., a word distribution), all smoothing methods attempt to generate a smoothed language model $P(w|d)$ that can better represent the topical content of document d. An obvious improvement over $P_{ML}(w|d)$ that almost all the smoothing methods would do is to avoid assigning zero probabilities to words that are not seen in d. Indeed, this was a major motivation for smoothing discussed in [139]. In general, however, the purpose of smoothing is to improve the accuracy of the estimated language model, not just avoiding zero probabilities. The most useful resources for smoothing so far have been documents that are similar to d, and methods exploiting such document graph structures are among the best performing methods [165, 79, 97].

The general procedure of all the smoothing methods using corpus structures is as follows:

1. Construct a graph of documents where documents are connected with edges weighted according to the similarity between them [1].
2. For every document, estimate a structure-based language model based on its nearest neighbors, or a cluster which that the document belongs to.
3. Combine the structure-based language model with the original unsmoothed document language model.

How are these methods different from traditional global methods, such as the Jelinek-Mercer(JM) and the Dirichlet smoothing [182]? Intuitively, they all went beyond the initial goal of giving non-zero probabilities to unseen words. But why do they all take such general steps? What are they essentially trying to optimize? To explain this formally, let us first look at what each step is trying to achieve.

[1] Although the graph structure is not explicitly mentioned in cluster-based methods [178, 97], this is a reasonable generalization.

By constructing a similarity graph of documents, one ensures that similar documents are located close to each other. Using the argument of data manifold in machine learning, if we project the documents onto a hyperplane, we expect that documents with the largest similarity have the smallest distance on the hyperplane [6].

By estimating a document language model based on the neighbor documents or the closest clusters, one ensures that the language model representation of a document is not significantly different from the documents close to it.

However, with this objective alone, the smoothed language model could dramatically deviate from the original document. Indeed, an easiest solution is making all documents share the same representation. The combination with original language model ensures that the smoothed language model is not far from its original contents.

Thus we see that there are two major intuitive assumptions made in all such work: "*documents in the same cluster should have similar representation*" [178, 97], and "*neighbor documents should have similar representation*" [79, 165]. Such assumptions also appear in the literature of semi-supervised learning [185, 189], where the first one is referred as "*global consistency*" and the latter as "*local consistency*".

Indeed, if we project the graph structure of documents on a hyperplane, the language model $\{P(w|d)\}$ can be plotted as surfaces on top of the hyperplane.

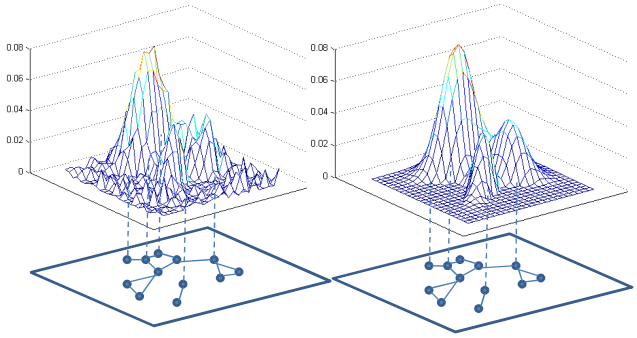


Figure 10.1: Illustration of smoothing language models on a graph structure. Left figure: unsmoothed model; Right figure: smoothed model

Figure 10.1 visualizes such an intuitive explanation with synthetic data. The hyperplane shows a manifold structure of documents and the surface plots $P(w|d)$ for a given word w over different documents. The left figure shows the maximum likelihood estimate of $P(w|d)$, which has an unsmoothed surface; in contrast, the figure on the right side presents a smoothed surface (i.e., smoothed $P(w|d)$) with the basic shape of the surface preserved (i.e., being "loyal" to $P_{ML}(w|d)$).

We can now see that graph structures are fundamental to smoothing and a better graph would presumably

lead to better smoothing. For example, the simple smoothing strategy of interpolating with the collection language model can be regarded as smoothing on a clearly non-optimal graph where document d is assumed to have equal distance to all other documents in the collection (i.e., a "star" structure). Thus it should not be surprising that smoothing on a graph better reflecting the real semantic similarities between documents would be better, which has indeed been shown in the literature [79, 165].

A major challenge in any smoothing method is how to deal with the tradeoff between "fidelity" (stay close to the non-smoothed model) and "smoothness" (be the same as neighbors). In general, we have such a surface for each word w' ($P(w'|d)$). Thus conceptually we have as many surfaces as the number of words in our vocabulary, and optimizing this tradeoff for all the words can be very difficult without an explicit objective function.

How can we ensure that the tradeoffs for different nodes in the graph are all reasonable? In general, how can we smooth all such surfaces in a principled way and achieve the smoothing effect in Figure 10.1? The hyperplane in Figure 10.1 illustrates a manifold of documents. Does that have to be a graph of documents? We can imagine that the hyperplane presents other types of graphs, such as a word graph, and plot the language models as different surfaces on that hyperplane. In the following section, we introduce a general framework to smooth language models on a general graph structure.

10.2.2 A General Framework

From Figure 10.1, we see that the notion of *smoothing* discussed in this chapter is different from "assigning non-zero probabilities to unseen words," but aims at "achieving consistency on a graph structure." For continuous functions like time series, smoothing is usually done with a regularizer with well-defined derivatives. A graph structure, however, defines a discrete domain. Discrete regularization has been cast as an optimization problem in machine learning literature [188]; we apply it here to define a general optimization framework for smoothing language models.

Formally, let us introduce the following definitions:

- $G = \langle V, E \rangle$: a graph defined in a retrieval problem. $u, v \in V$ are vertices and $(u, v) \in E$ is an edge.
- f_u: a smoothed value-based representation of vertex u in the graph G (e.g., $P(w|d_u)$).
- $\tilde{f}_u$: a non-smoothed (initial) value of u (e.g., $P_{ML}(w|d_u)$).
- $w(u)$: a weight of the importance of vertex u in G.
- $w(u, v)$: a weight of the importance of edge (u, v).

We then denote $\mathcal{D}$ as the text collection, D as the set of documents and W as the set of words. Note that G can be either directed or undirected. In this chapter, we only focus on the undirected case, and propose the following general optimization framework for smoothing language models on the graph structure.

$$\begin{aligned} O(\mathcal{D}) &= (1-\lambda)\sum_{u\in V} w(u)(f_u - \tilde{f}_u)^2 \\ &+ \lambda \sum_{(u,v)\in E} w(u,v)(f_u - f_v)^2 \end{aligned} \tag{10.1}$$

This objective function generalizes the intuitions in Section 10.2.1 well: The first term guarantees that the smoothed language model does not deviate too much from its original value, especially for more important vertices (controlled by $w(u)$); the second term, also known as a harmonic function in semi-supervised learning, guarantees the consistency of the language model on the graph. The right term can also be written as $\lambda \sum_{u\in V}\sum_{v\in V} w(u,v)(f_u - f_v)^2$ where $w(u,v) = 0$ when there is no edge between u, v.

This framework is general. Clearly, a different selection of $\{G, f_u, \tilde{f}_u, w(u), w(u,v)\}$ leads to a different smoothing strategy. This flexibility provides us a road map for understanding the existing smoothing strategies, as well as exploring novel smoothing methods.

To use such a road map, any reasonable instantiation of the framework must satisfy the following constraints.

1. f_u and f_v are comparable, that is, when f is fully smoothed on G, f_u and f_v should be the same value.
2. $w(u)$ offers a reasonable weighting that captures the importance of vertex u in G.
3. $w(u,v)$ offers a reasonable weighting that captures the importance of edge (u,v) in G.

When we are smoothing language models, we could instantiate f_u as the probability of a word given a document, i.e., $P(w|d)$, and associate u with d, or w, or both. In this case, there is an additional constraint, i.e., $\sum_w P(w|d) = 1$.

What are reasonable instantiations of $w(u)$ and $w(u,v)$? Intuitively, $w(u,v)$ could be instantiated as the closeness of two vertices in the graph, or the similarity of u and v. Using the similarity of two vertices to weight an edge has been commonly adopted in existing literature ([79, 165]). How about $w(u)$? There are also many studies in the context of link analysis to model the importance of a vertex on a graph, e.g., in-degree, PageRank [14], and HITS [73]. In this chapter, we use the degree of vertex u as the importance weight of u:

$$w(u) = Deg(u) = \sum_{v\in V} w(u,v) \tag{10.2}$$

Minimizing $O(\mathcal{D})$ in Equation 10.1 will achieve the smoothness of the surface in Figure 10.1. For instance, if we select $\{G = D, f_u = P(w|d_u), \tilde{f}_u = P_{ML}(w|d_u), w(u,v) = Cosine(d_u, d_v)\}$, the smoothing framework boils down to smoothing language models with document similarity graph, where edges are weighted with cosine similarity. To smooth the language models $\{P(w|d)\}_{d\in\mathcal{D}}$, one needs to find a solution of $\{P(w|d_u)\}_{d_u\in\mathcal{D}} = \arg\min_{\{f_u\}_{u\in V}} \sum_w O_w(\mathcal{D})$, subject to the constraint $\sum_w P(w|d) = 1$, which achieves the smoothness of multiple surfaces.

10.2.3 Connection to NetPLSA

It is not hard to show that the proposed optimization framework is a special case of the general contextual topic model discussed in Chapter 4. In fact, it is a special case of the NetPLSA model we proposed in Chapter 7, specifically a contextual language model with graph-based regularizer. Indeed, no matter whether f_u in Equation 10.1 corresponds to a document language model or a query language model, it is simply a part of a word distribution corresponding to the context model of either a document or a query. Since the document contexts and query contexts do not overlap with each other, there is only one possible view, a one possible coverage, and one possible topic for each document/query - the document/query itself. Therefore, if we apply the maximum likelihood estimator to estimate f_u, we will get $\tilde{f}_u$. So the left component of the objective function in Equation 10.1 is equivalent to maximizing the likelihood of the contextual language model with documents/queries as contexts, which is a special case of the CPLSA model. The right component is a graph-based regularizer on the model parameters, like in NetPLSA.

What is interesting here is that the graph structure could either be a graph of documents or a graph of words. If a document graph is adopted, the regularizer is attempting to model the dependency between contexts; if a word graph is adopted, the regularizer is attempting to model the prior knowledge about the connections between language units.

10.2.4 The Smoothing Procedure

Generally, to minimize $O(\mathcal{D})$ in Equation 10.1, we can compute the first-order partial derivatives of $O(\mathcal{D})$, that is,

$$\frac{\partial O(\mathcal{D})}{\partial f_u} = 2(1-\lambda)Deg(u)(f_u - \tilde{f}_u) + 2\lambda \sum_{v\in V} w(u,v)(f_u - f_v)$$

Let $\frac{\partial O(\mathcal{D})}{\partial f_u} = 0$ and plug in Equation 10.2, we have

$$f_u = (1-\lambda)\tilde{f}_u + \lambda \sum_{v \in V} \frac{w(u,v)}{Deg(u)} f_v \tag{10.3}$$

Clearly, a solution of Equation 10.3 could minimize $O(\mathcal{D})$ in Equation 10.1. To get such a solution, we can start with $f_u = \tilde{f}_u$ and execute Equation 10.3 until converging. In practice, we do not need to wait until complete convergence; a few iterations of Equation 10.3 can already give an improved $O(\mathcal{D})$. We further discuss leave convergence in Section 10.4.

10.3 Instantiations on Document and Term Graphs

In Section 10.2, we have shown that smoothing document language models with a document graph [97, 79, 165] is an instantiation of the general framework. It is by no means the only one. Indeed, any reasonable instantiation of $\{G, f_u, \tilde{f}_u,\ w(u), w(u,v)\}$ which satisfies the predefined constraints will lead to a variant strategy of smoothing. In this section, we explore different instantiations of the framework.

10.3.1 Smoothing with a Document Graph

The most common instantiation is smoothing with document graphs (i.e., $V = D$). Although not in a unified way, many heuristics have been proposed to smooth using document graphs. Document language models are adjusted by receiving weights from the cluster it belongs to [178, 97], or by propagating weights from the nearest neighbors [79, 140, 165, 152]. A commonly used instantiation of $w(u,v)$ is the cosine similarity of two documents. As shown in Section 10.2.2, a reasonable instantiation of f_u and $\tilde{f}_u$ is $f_u = P(w|d_u)$ and $\tilde{f}_u = P_{ML}(w|d_u)$. Plugging these in Equation 10.3, we have

Smooth Document Language Models ($f_u = P(w|d_u)$):

$$P(w|d_u) = (1-\lambda)P_{ML}(w|d_u) + \lambda \sum_{v \in V} \frac{w(u,v)}{Deg(u)} P(w|d_v) \tag{10.4}$$

One may notice that we did not utilize the constraint $\sum_w P(w|d) = 1$. As a matter of fact, when we start with $P(w|d_u) = P_{ML}(w|d_u)$, this constraint is naturally satisfied after every iteration of Equation 10.4.

Another interesting instantiation of f_u is simply the relevance score of a document d_u for a query q.

Smooth Relevance Scores ($f_u = s(q, d_u)$):

$$s(q,d_u) = (1-\lambda)\tilde{s}(q,d_u) + \lambda \sum_{v \in V} \frac{w(u,v)}{Deg(u)} s(q,d_v) \tag{10.5}$$

This process bypasses language models, but smoothes the relevance score directly. Similar heuristics appear in existing work where corpus structure is utilized to rerank documents with score propagation and regularization [80, 38, 140, 81]. In our experiments, we will show that this instantiation is not as effective as smoothing document language models.

10.3.2 Smoothing with a Word Graph

Recent work in natural language processing has introduced new ways to represent documents with graph structures of words [119]. Indeed, in many scenarios we are accessible to a graph of terms (e.g., a word similarity graph, an entity-relation graph, or an ontology graph). Although most existing work smoothes language models with a document graph, we show that the general smoothing framework can also be instantiated with word graphs, which leads to novel smoothing procedures.

In such smoothing procedures, G is now a word graph (i.e., $V = W$). $w(u, v)$ can be instantiated with either co-occurrence or mutual information of two words. We introduce the following novel instantiations:

Smooth Document Language Models ($f_u = \frac{P(w_u|d)}{Deg(u)}$):

$$P(w_u|d) = (1 - \lambda)P_{ML}(w_u|d) + \lambda \sum_{v \in V} \frac{w(u, v)}{Deg(v)} P(w_v|d) \tag{10.6}$$

Note that unlike a document graph where all documents are treated equally, there is significant prior knowledge of using different words. Some words tend to be used more than others even if they are highly related (e.g., car v.s. vehicle). Some words are more important than others and thus should be assigned a larger value. In this instantiation, we use $f_u = \frac{P(w_u|d)}{Deg(u)}$ rather than $P(w_u|d)$ to make f_u and f_v comparable. Please note that $Deg(u)$ is just one choice of denominator which captures the importance of a word (which is proportional to the pagerank value on G). One could use other choices such as idf. This results in a different denominator from the one used in Equation 10.4 and Equation 10.6 ($Deg(u)$ v.s. $Deg(v)$). $\sum_w P(w|d) = 1$ is now naturally satisfied after any iteration of Equation 10.6. This smoothing strategy is not well studied in existing IR literature. Interestingly, if we use a similar instantiation, $f_u = P(d_u|w)/Deg(u)$ on a document graph based on document similarity, we would be able to derive the term count propagation smoothing method proposed in [152].

The instantiations discussed so far aim at smoothing document language models, where $f_u \propto P(w_u|d)$. When systematically examining variations of the framework, one may naturally ask whether this is the only way to instantiate f_u. In language modeling retrieval models, the query language model is also an important component. Can we also leverage the framework to smooth query language models? The answer is yes. Let

$f_u = \frac{P(w_u|q)}{Deg(u)}$, we have

Smooth Query Language Models ($f_u = \frac{P(w_u|q)}{Deg(u)}$):

$$P(w_u|q) = (1-\lambda)P_{ML}(w_u|q) + \lambda \sum_{v \in V} \frac{w(u,v)}{Deg(v)} P(w_v|q) \tag{10.7}$$

This smoothing method is related to query expansion (i.e., to add new terms to the query and adjust the weights of query words). In model-based feedback [181], where a new query model is estimated from feedback documents, we expect that this novel smoothing strategy can also be applied.

10.4 Discussion of the Framework

The objective function of smoothing in Equation 10.1 is related to existing work of discrete regularization in semi-supervised learning and manifold learning [189, 6, 185, 188]. Many different regularizers and objective functions have been proposed in that context. We use Equation 10.1 instead of others because it is general, has many nice properties, and has natural connections to many existing concepts and models. Some recent work [38] also used one of them for the problem of retrieval score regularization.

10.4.1 Connection with Existing Models

Equation 10.6 and Equation 10.7 look similar to the updating formula of PageRank [14], except that $P_{ML}(w|d)$ is used instead of the uniformed jumping probability $1/N$. Indeed, Equation 10.6 can be rewritten as

$$P(w_u|d) = \sum_{v \in V} ((1-\lambda)P_{ML}(w_u|d) + \lambda \frac{w(u,v)}{Deg(v)}) P(w_v|d), \tag{10.8}$$

which is essentially computing a stationary distribution of a positive-recurrent Markov chain, where

$$p(v \rightarrow u) = (1-\lambda)P_{ML}(w_u|d) + \lambda \frac{w(u,v)}{Deg(v)},$$

which is guaranteed to converge. Note that the initial value of u would not be forgotten like in PageRank, because $P_{ML}(w_u|d)$ has been embedded into transition probabilities. Intuitively, we can imagine that when an author is composing a document, he would randomly walk along such a Markov chain of words, and write down a word whenever he passes it.

PageRank-like propagation has been used in [80, 140] to rerank top retrieved documents, combined with other features in a heuristic way.

What about Equation 10.4 and 10.5 for the document graph, or more generally Equation 10.3? They are not like PageRank now because a different denominator $Deg(u)$ is used instead of $Deg(v)$. What is this essentially modeling? In fact, when we transform the graph in a certain way (by adding nodes and reassigning transition probabilities), one could see that Equation 10.4 is actually computing the "absorption probability" – the probability of each node to be absorbed to a termination state in such a transformed Markov chain. This is also guaranteed to converge.

Note that if we only apply Equation 10.4 once, the smoothing procedure is very similar to the method proposed in [165]. The difference is that they are using $f_u = c(w, d_u)$, which we have shown to be not as reasonable as $P(w|d_u)$.

How about language model smoothing with collection distribution (e.g., Jelinek-Mercer smoothing) and cluster structures [97, 81]? Intuitively, these models are smoothing document language models with global structures (e.g., collection, clusters), so that the estimated language models could satisfy global consistency (i.e., documents in the same global structure has similar representation). [165, 79] explored nearest neighbors, which guarantees local consistency of language models. A regularization framework like Equation 10.1, as shown in [185], satisfies both local and global consistency on a manifold structure.

10.4.2 Selection of Graphs

The smoothing framework we proposed is general and orthogonal to the selection of graphs. As long as the graph is constructed in a reasonable way (two related vertices are connected with a higher weighted edge, and are expected to have similar representations), the objective function in Equation 10.1 can be applied.

Thus either a fully connected graph (with well scaled edge weights) [185, 188], or a k-nearest-neighbor (kNN) graph [79, 80, 165] can be used for smoothing. We show our experimental results with kNN graphs. On the other hand, any reasonable distance/similarity measure could be applied to compute $w(u, v)$.

In scenarios that the number of documents/terms is too large, as in the case of a commercial search engine, one may think of using smaller subgraphs. Our proposed framework can be easily adapted to subgraphs. Of course, we need to ensure that the smoothed value of vertices in the subgraph are comparable with the values of the unsmoothed vertices (i.e., vertices not in the subgraph). The representations of both smoothed documents and unsmoothed documents with Equation 10.4 are all language models, which do not have a scaling problem. As for term graphs (e.g., Equation 10.6), it is easy to prove that

$$\sum_u P(w_u|d) = \sum_u P_{ML}(w_u|d).$$

This formula indicates that the probability of smoothed terms would not affect the probability mass of

unsmoothed terms.

10.5 Experiments

In Section 10.2.2 and Section 10.3, we proposed a general framework of smoothing language models and introduced various instantiations with document graph and word graph, resulting in several different smoothing strategies. In this section, we evaluate the effectiveness of these strategies empirically. Experiment results show that all the instantiations of the smoothing framework outperform the non-structural smoothing methods. Two instantiated approaches also outperform the state-of-the-art graph-based smoothing methods.

10.5.1 Experiment Setup

We use four representative TREC data sets: AP88-90, LA (LA Times), SJMN (San Jose Mercury News 1991), and TREC8. They are identical to the data sets used in [165], with the same source, query, and preprocessing procedure. The first three data sets are also identical to the data sets used in [97]. The basic statistics of the data sets are presented in Table 10.1. We used the title field of a query/topic description to simulate short keyword queries in our experiments.

	#documents	avg dl	queries	#total qrel
AP88-90	243k	273	51-150	21819
LA	132k	290	301-400	2350
SJMN	90k	266	51-150	4881
TREC8	528k	477	401-450	4728

Table 10.1: Basic information of data sets

For every data set, we construct a k-Nearest-Neighbor (kNN) graph of all documents, as well as a kNN graph of words. The edge weight of two documents is measured with the cosine similarity, formally

$$sim(d_1, d_2) = \frac{\sum_w c(w, d_1) \times c(w, d_2)}{\sqrt{\sum_w c(w, d_1)^2 \times \sum_w c(w, d_2)^2}}$$

.

The edge weight of two words is set to their mutual information [31]. We do not include the most frequent terms (which appear in $> 50\%$ documents) or the most infrequent terms (which appear in < 15 documents in TREC8, or < 7 documents in other data sets). This provides us with a word graph of $40k \sim 60k$ vertices. We control the density of the graph by adjusting the number of nearest neighbors. To ensure that the graph is undirected, we add an edge between u and v if either u is in v's k-nearest neighbors, or the converse.

After we smooth all the document language models and obtain the smoothed $P'(w|d)$ and possibly also a smoothed query language model $P(w|q)$, we use further smooth $P'(w|d)$ using Dirichlet prior [184] and obtain

$$P''(w|d) = \frac{|d|}{|d|+\mu}P'(w|d) + \frac{\mu}{|d|+\mu}P(w|C)$$

We then use the KL-divergence retrieval model to rank documents, where each document is scored based on the negative KL-divergence of the query language model and $P''(w|d)$ [82].

The additional Dirichlet smoothing is to model noise in the query and has been used in most existing work on smoothing language models with corpus structure [97, 79, 165]. When $P'(w|d)$ is unsmoothed (i.e., the maximum likelihood estimate), it boils down to the Dirichlet prior model, which is used as our baseline.

10.5.2 Basic Results

In Section 10.3, we introduced four different instantiations of smoothing, namely: smoothing document language model with document graph (**DMDG**); smoothing relevance score with document graph (**DSDG**); smoothing document language model with word graph (**DMWG**); and smoothing query language model with word graph (**QMWG**).

Data		Dirichlet	DMDG	DMWG†	DSDG	QMWG
AP88-90	MAP	0.217	0.254 (+17.1%***)	0.252 (+16.1%***)	0.239 (+10.1%***)	0.239 (+10.1%)
	pr@10	0.432	0.447 (+3.5%*)	0.461 (+6.7%***)	0.453 (+4.9%**)	0.451 (+4.4%)
LA	MAP	0.247	0.258 (+4.5%**)	0.257 (+4.0%**)	0.251 (+1.6%**)	0.247 (0.0%*)
	pr@10	0.287	0.290 (+1.0%)	0.301 (+4.8%*)	0.285 (-0.7%)	0.287 (0.0%)
SJMN	MAP	0.204	0.231 (+13.2%***)	0.229 (+12.3%***)	0.225 (+10.3%***)	0.219 (+7.4%)
	pr@10	0.298	0.320 (+7.4%***)	0.326 (+9.3%***)	0.329 (+10.4%***)	0.326 (+9.4%)
TREC	MAP	0.257	0.271 (+5.4%***)	0.271 (+5.4%**)	0.261 (+1.6%)	0.260 (+1.2%)
	pr@10	0.450	0.468 (+4.0%*)	0.466 (+3.6%*)	0.464 (+3.1%)	0.474 (+5.3%**)

Table 10.2: Basic Results: Graph-based smoothing outperforms non-structural smoothing
†For efficiency reason, we conduct the DMWG smoothing by reranking the top 3000 results returned by Dirichlet method. All other methods are applied on the complete set of documents. We measure the statistical significance of the improvement using Wilcoxon test. *,**,*** means that the improvement hypothesis is accepted at significance level 0.1, 0.05, 0.01 respectively.

We compare these four methods in Table 10.2 along with the best results of the Dirichlet smoothing. In all our experiments, the cutoff of relevant documents is set as 1000. We see that all the four smoothing instantiations outperform the non-structural smoothing baseline. DMDG and DMWG outperform Dirichlet prior consistently and significantly.

Among the four proposed methods, we see that smoothing document language model tends to achieve better performance than smoothing query language model or the relevance score. One possible explanation is that smoothing document models is superior to smoothing query language models, since that the short

keyword query only conveys very sparse information about the user's information need. Expanding a query in a wrong direction could hurt the retrieval performance.

Regularizing relevance scores has been discussed in existing literature [38], where a similar approach to DSDG is used. Clearly, we see that smoothing relevance scores alone is not as effective as smoothing the document language models. Indeed, by dealing with the richer representation of document language models, one has more flexibility in controlling the core retrieval modules to achieve better performance.

Using a document graph vs. a query graph to smooth document language models perform similarly, which is appealing since smoothing with word graph has not been well explored in the existing literature, suggesting potential room for further leveraging a word graph for smoothing.

10.5.3 Tuning Parameters

The selection of smoothing strategies would introduce a different set of new parameters. Generally, when a kNN graph is used, the size of the graph could be controlled by tuning the parameter k. In Equation 10.1, we introduced a parameter λ to balance the consistency of the language models on the graph, and the fidelity to the maximum likelihood estimates. Figure 10.2 presents the sensitivity of the retrieval performance to these introduced parameters.

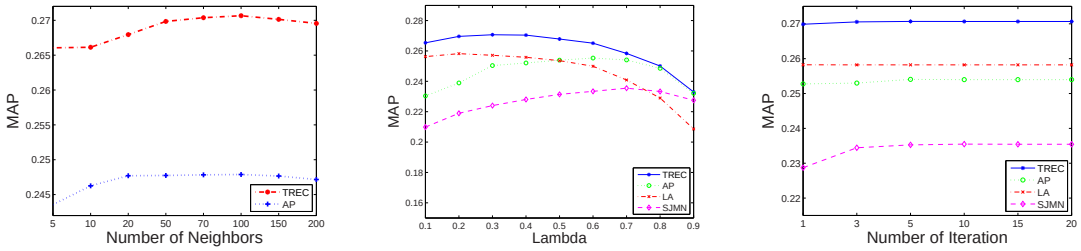


Figure 10.2: The sensitivity of parameters (k, λ, and iteration)
All results are obtained by DMDG. $k = 100$, $iteration = 10$ unless specified. Similar patterns are observed from other methods.

From the left plot, we see that the performance is relatively stable over different k, even if a document only has a few (e.g., 10) neighbors. This is different from the observation made in [165], where smoothing with a larger number (≥ 100) of neighbors significantly outperforms a small number (≤ 50) of neighbors. This is because when propagations are processed iteratively, a few closest neighbors could well capture the local consistency.

Similar patterns are observed from other data sets and smoothing methods. We set the number of neighbors to 100 for a document graph and 50 for a word graph, unless specifically noted.

From the plot in the middle of Figure 10.2, we see that when λ is larger, the estimated language models

would achieve more consistency on the surface at the expense of deviating more from the original estimates. When λ is set smaller, less smoothing effect is added to the language models. Setting λ in the range of $0.3 \sim 0.7$ usually yields good retrieval performance.

Finally, how many iterations do we have to run for the equations proposed in Section 10.3? In fact, we could even find a closed form solution for such a regularizer [185, 38]. However, its computation is time consuming. Intuitively, most smoothing effect occurs in the first a few iterations, which is indeed confirmed in the right plot of Figure 10.2, where we see that the performance becomes quite stable after a few iterations. Thus we may just run the algorithm for a few iterations in practice.

10.5.4 Comparison with Existing Methods

We now compare our methods with some state of the art smoothing methods.

	CBDM	DELM	DMDG	DMDG (1 iter.)
AP88-89	0.233	0.250	**0.254**	0.252
LA	0.259	**0.265**	0.260	0.258
SJMN	0.217	0.227	**0.235**	0.229
TREC	N/A	0.267	**0.271**	0.270

Table 10.3: Performance (MAP) comparison with existing smoothing methods

Liu and Croft [97] proposed a method (denoted as CBDM) to smooth document language models with cluster language models. [165] also proposed a smoothing strategy (denoted as DELM) to expand a document with its nearest neighbors. Both methods utilized the document similarity and are related to the DMDG instantiation we proposed. As we use the identical data sets and preprocessing procedures with their work, we compare the performance of our DMDG method with the best results reported in their papers.

We see that DMDG consistently outperforms CBDM. It also outperforms DELM on three data sets except for LA, but with a smaller improvement. This is not surprising, because the DELM method is very similar to our DMDG instantiation if only one iteration of Equation 10.4 is processed. The difference is that DELM also expands the document length besides smoothing the language models. The performance of DMDG with only 1 iteration is also presented in Table 10.3, from which we see that running more iterations improves retrieval performance. Similar improvements are achieved with the DMWG method. This comparison shows that the optimization framework of smoothing performs better than exploring either local or global consistency alone.

10.5.5 Combination with Pseudo-feedback

Pseudo feedback has been proved to be effective to update the query model using collection information [181]. It is interesting to see whether pseudo feedback could bring additional improvements to the graph-based smoothing methods, and whether it could benefit from a graph structure.

[165] has already proved that combining pseudo feedback and document graph achieves better results than both the feedback model and the smoothing model alone. In this work, we intend to explore whether combining the feedback model and the word graph would improve performance, because smoothing with a word graph is our novel exploration. Model-based feedback [181] combines the original query language model with a feedback model estimated from the top ranked documents. One natural thought is to smooth the feedback model with a word graph and Equation 10.7. An alternative way is to use the updated query model to retrieve the smoothed documents (i.e., with DMWG method) while the document language models are smoothed. We explored both strategies, and summarize the results in Table 10.4.

	FB	FB+QMWG	DMWG	FB†	FB+DMWG
AP	0.271	**0.273**	0.252	0.266	**0.271** **
LA	0.258	**0.267**	0.257	0.257	**0.267** **
SJMN	0.245	**0.246**	0.229	0.241	**0.249** **
TREC	0.278	**0.280**	0.271	0.278	**0.292** ***

Table 10.4: Performance (MAP) of combination of word graph and pseudo feedback
†to be consistent with DMWG, we use the same setup feedback (rerank the top 3000 docs and output 1000). This usually yields to a reduced performance than ranking all the documents.

We use the model-based method in [181] to estimate a feedback model from the top 5 retrieved documents. In this process, we fix the noise parameter to be 0.9. We tune the combination parameter α (as in [181]) to get the optimal performance of pseudo-feedback. From Table 10.4, we see that both ways of combination improved performance. Combining pseudo feedback and DMWG significantly improves both DMWG and pseudo feedback. For the FB+DMWG model, we simply reuse the optimal parameters for Feedback and DMWG individually, without further tuning.

10.6 Related Work

Most of the related work has already been discussed in the previous sections of the chapter. Here we give a brief summary of all the related work.

Smoothing language models [23] for information retrieval has been a fundamental and challenge problem in IR research. Traditional smoothing methods explore the collection information in an unstructured way [139, 23, 182]. Our work lies in the context of language model smoothing, but we utilize the graph structures

from the collection to smooth document and query language models.

Recent research has explored document similarity graphs to smooth language models [178, 97, 79, 140, 165]. Cluster structures [178, 97], nearest neighbors [79, 165], and propagation-based methods [140, 81] have been proposed with various heuristic solutions. Our proposed optimization framework is a reasonable generalization of all such approaches, which provides a unified solution to this problem, as well as a road map for exploring novel smoothing approaches.

One of the concrete instantiations of our proposed framework, smoothing relevance scores with document graph, is related to score reranking work, such as [80, 38, 81]. [38] explores a regularization approach which is related to the objective function we introduced. However, they focus on regularizing the relevance scores, while we explore the general problem of smoothing language models. Experimental results show that smoothing relevance score alone is not as effective as smoothing document language models.

The proposed optimization framework is also related to the graph-based learning theme in machine learning. Similar objective functions have been explored in [189, 185, 188]. Their main focuses are machine learning problems such as semi-supervised learning and spectral clustering, while we explore graph structures to smooth language models for retrieval.

10.7 Summary

In this chapter, we proposed a general optimization framework for smoothing language models with graph structures. The proposed framework not only gives a principled justification for many of the heuristics and a unified solution to smoothing language models with graphs, but also provides a road map for the exploration of novel smoothing methods. Following such a road map, we introduced four instantiations of smoothing methods, including two novel methods of smoothing document and query models with a word graph. Empirical results show that all proposed instantiations significantly improve over the unstructured smoothing methods. The two methods of smoothing document language models outperform the state of the art smoothing methods.

As we have discussed, a specific characteristic of text data is that the data is very sparse. A good generative model for the a collection of text data should take this problem into consideration. In fact, it is common practice in speech recognition and natural language processing to alleviate this data sparsity problem, by smoothing and regularizing the text models [68, 46, 22]. In contextual text mining, when we drill down to individual contexts, the data sparsity problem becomes an even more serious concern. In information retrieval, for example, the performance is closely related to how the document language model

is constructed, which further boils down to how the document language model is smoothed.

Think about the more general contextual mixture model (Equation 3.2). When we have too many free parameters to estimate, the model is likely to overfit the sparse data. The basic approach to avoid this is to add constraints to the model parameters, so that the number of parameters is reduced (e.g., PLSA, CPLSA, TSM), or add regularizers of the the model parameters, so that the value of the parameters follow some criteria, or some prior distribution (e.g., LDA, TSM). In Chapter 7, we proposed a novel approach to regularize the model parameters with a discrete regularizer based on a graph structure of contexts. The advantage of this approach is that we can incorporate an explicit objective function for the smoothing of the model parameters. Unlike regularization with a prior distribution, which tries to "drag" the parameters towards the mode of that prior distribution, a discrete regularizer treats different parameters differently, so that they achieve a "harmony" on a graph structure.

This chapter illustrates a good example of utilizing the techniques of contextual text mining to facilitate text information management tasks. We use as an example the smoothing of language models in text retrieval. A simple instantiation of the contextual language model, with the focus on how to use the graph-based regularization to smooth the model parameters. The graph-based smoothing framework we presented is a special case of the NetPLSA model proposed in Chapter 7. Empirical experiments using standard TREC datasets prove that the techniques of contextual text mining work effectively on the task of document retrieval in a quantitative way.

Chapter 11

Application: Contextual Text Mining in Web Search

In the previous chapter, we illustrated an application of contextual text mining in the "context" of text retrieval, the goal of which is to smooth the document and query language models based on the dependency structure of documents and terms. In this chapter, we introduce another interesting application of contextual text mining in the "context" of web search. Specifically, we look at the search logs collected by a commercial search engine as our text data, and study how the modeling of users as context can help with web search. Another motivation of this work is to test whether contextual text mining models can effectively process very large scale data sets (e.g., the web scale).

In particular, with the modeling of the context of users, we can provide personalized search to particular users. We further propose a novel personalization model, called personalization with backoff, to smooth the context models of particular users based on nested groups of users close to the target user (see [108]). The personalization with backoff model can be shown as a special case of the CPLSA model. Empirical experiments show that the modeling of user context in web search is very effective. It can potentially cut the difficulty of search by half. We further show that by incorporating other types of contexts into web search, such as days-of-the-week, time-of-the-day, and types of queries, there is considerable potential to further improve the effectiveness of web search.

11.1 Overview

How many pages are there on the Web? 5B? 20B? More? Less? How hard is search? How much does personalization help? All are difficult but crucial questions to search business.

Scale is hard. The bigger the web, the harder the search. Search engines make large investments in expensive computer centers in the cloud to index billions of pages. Could these large investments be wiped out if a small cache of a few million pages could capture much of the value? What if someone found a way to squeeze much of the value of the cluster into a desktop or a mobile device? Is search more like an Everest expedition (clusters in the clouds) or a walk in the park (a little flash memory on a mobile device)?

Related questions come up in language. How big is English? One can find simple answers on the covers of many dictionaries, but we would feel more comfortable with answers from a more authoritative source than a marketing department. Many academics have contributed to this discussion from many perspectives: Education, Psychology, Statistics, Linguistics, and Engineering. Chomsky and Shannon proposed two different ways to think about such questions:

- Chomsky: language is infinite [25]
- Shannon: 1.3 bits per character [154]

These two answers are very different. Chomsky's answer is about the total number of words; and Shannon's answer is about the perplexity, or the difficulty of using a language. A dictionary could cover a lot of words, but not all of them are actively used. Using a Chomskian argument, we could argue that there are infinitely many urls. For example, one could write a spider trap such as successor.aspx?x=0 which links to successor.aspx?x=1 which links to successor.aspx?x=2. In addition to intentionally malicious spider traps, there are perfectly benign examples such as calendars[1], where there are infinitely many pages, one for each month, with links from each month to the next. It is all too easy to build a web crawler that finds itself attempting to materialize an infinite set with finite resources. The crawler can easily consume all available time and space.

Shannon offers a more practical answer. Although there are a lot of pages out there, there are not that many pages that people actually go to. This chapter will estimate entropy of urls (and queries and IP addresses) based on logs from Microsoft's www.live.com. We find that it takes just 22 bits to guess the next url (or the next query or the next IP address). That is a walk in the park (millions, not billions). With all the talk about the long tail, one would think that the web was astronomical. But the logs are tiny, far less than Carl Sagan's billions and billions [151].

As we will see, entropy is a powerful tool for sizing challenges and opportunities. By estimating the entropy from search logs, we expect to answer key questions in search business: How hard is search? How much does it help if we know the context of the user?

Personalization

Personalization is a hot topic, with a large body of work, not only in the scientific literature, but also in commercial practice. Many people use personalized search products every day. A query for "personalized search" returns millions of page hits. The first few pages of results are dominated by the commercial practice.

[1] http://www.timeanddate.com/calendar/monthly.html?year=2005&month=12&country=11

If you want to find the scientific literature such as [166], you'll have to refine the query considerably by adding a keyword like "SIGIR."

Why does personalization help? It is useful to know your audience. Consider the ambiguous query: "MSG". Depending on the user, this query could be looking for the sports arena (Madison Square Garden) or the food additive (Mono-sodium Glutamate). The search engine could do a better job answering ambiguous queries like this if it had access to demographic data and/or log data such as click logs.

Many acronyms are ambiguous. ACS can refer to the "American Chemical Society," the "American Cancer Society," the "American College of Surgeons" and more. Acronyms take on special meanings inside many large organizations and private enterprises. For example, for most people, MSR means "Mountain Safety Research," but inside Microsoft, it means "Microsoft Research." And of course, it means other things to other people including: "Montessori School of Raleigh," "Mom Service Representative" and "My Sports Radio." PSS is a stock ticker for "Payless Shoes," as well an abbreviation of several different companies: "Physicians Sales and Service," "Phoenix Simulation Software," "Personal Search Syndication," "Professional Sound System," etc. But inside Microsoft, PSS refers to "Product Support Services." It helps to know your audience in order to know:

- what the terminology means
- which questions are likely to come up, and
- which answers are likely to be appreciated.

In our perspective, personalization refers to a particular instantiation of contextual text mining, where the a context denotes a user. If we have the relevant data (such as click logs) for a particular user, we should use it.

Personalization With Backoff

But what if we do not have data for a particular user (or we cannot use it because of privacy concerns)? This chapter takes a backoff approach to personalization [68]. If we do not have data for a particular user, back off to larger and larger groups of similar users. Under the framework of contextual text mining, we include both individual users and groups of similar users as contexts. As a proof of concept, users are grouped into equivalence classes based on the most significant bytes of their IP address. Personalization is then conducted by combining estimates based on five levels of contexts: all four bytes of the IP address, the first three bytes, the first two, and so on. It would be even better to group customers by market segments

and/or collaborative filtering (users who ask similar questions and click on similar urls). We leave these suggestions for future work.

We find that a little bit of personalization is better than too much or too little. Specifically, personalization with backoff to higher bytes of IP addresses (especially the second and third bytes) is better than 100% personalization or no personalization. Too little personalization misses the opportunity and too much runs into sparse data (and privacy). It isn't feasible to know everything about everyone (and they might not like it, if we knew too much).

Instead of assigning each customer to his own class (i.e., 100% personalization), it is common to assign customers to market segments. Market segments are typically defined in terms of context features such as geography (e.g., zip code), and time of day and day of week. These context features are easy to work with, and hopefully, they are well correlated with the more sensitive demographic variables such as those mentioned above.

11.2 Entropy Estimation

How big is the web? How hard is search? How much does personalization help?

To answer these questions and more, we collected a sample of logs from the Live search engine of about 1.5 year up to July 2007. This 1.5 year data, denoted as the "bigger" dataset, contains 193 million unique IP addresses, 637 million unique queries, and 585 million unique urls. The sample contains about 10 million $\langle Q, URL, IP \rangle$ triples per day. Each triple corresponds to a click from a particular IP address on a particular url for a particular query.

We separated the logs between 1/1/2006 and 2/6/2006 specifically for personalization experiments. The January data (the "smaller" data) was used for training the model of personalization and the February data was used for validation and testing. This one month training set contains 26 million unique IP addresses, 36 million unique queries, and 63 million unique urls. Entropy was estimated based on both the smaller data and the bigger data.

We assume that these sets are reasonably representative of the tasks of interest, though of course, such assumptions can be highly problematic. It is possible, for example, that users could share the same IP address, and a user would click on different urls under different conditions.

11.2.1 Notation

- **Q**: a query

- **U**: a user
- **IP**: an IP address. IP will be used as a convenient surrogate for U, though of course, it is possible for multiple users to share the same IP address.
- **C**: a class of users. Users are grouped into equivalence classes based on variables such as IP prefixes, time and location. These variables are treated as convenient surrogates for variables of interest such as demographics, market segments, etc.

  In this chapter, both Q, U, and C are context features that we select for the mining task. The value of each context feature can define the contexts q, u (ip), and c, corresponding to the conditions "the query is q," "the query is submitted by user u (from the IP ip)," and "the user belongs to group c."
- **URL**: Uniform Resource Locator, the name of a web document. In this chapter, a URL is our basic language unit. We use url to denote a particular value of URL.

  Please note that a query q can also be regarded as the language unit in the task of query suggestion. In the task of URL search, however, q refers to a context.
- $\langle Q, URL, IP \rangle$: a triple from the search logs, indicating that there was a click on a particular URL in response to a particular query Q from a particular IP address. All the triples $\langle q, url, ip \rangle$ consists our contextualized text collection.

11.2.2 Entropy (H)

Entropy[2] is commonly used in information theory to characterize the size of the search space. The larger the entropy of a distribution is, the harder it is to predict the next event [153]. In [154], Shannon used entropy to measure the difficulty of predicting the next character of English. Shannon's entropy provides bounds, but does not say how to achieve these bounds.

Similarly, we introduce entropy to measure the difficulty faced by a search engine. Note that entropy measures the size of the search space of the web, but not the number of particular urls. This is practical since what a search business cares about is the difficulty of search. How many bits does it take to guess the next url that will be clicked on?

$$H(URL) = -\sum_{url} p(url) \log p(url)$$

Conditional entropy measures the remaining entropy of the target random variable given the value of another related random variable. We can thus use conditional entropy to measure the difficulty of web

[2] http://en.wikipedia.org/wiki/Information_entropy

search, when we know the context of query, user, etc. The search task, of course, is much easier, because we are given the query, which is a huge hint.

$$H(URL|Q) = H(URL, Q) - H(Q)$$

How much does personalization help? That is, suppose we give the search engine not only the context of query, but also the context of IP address. How much does the IP address help?

$$H(URL|Q, IP) = H(URL, Q, IP) - H(Q, IP)$$

These quantities and more can be estimated from the training data, a sequence of triples: $\langle Q, URL, IP \rangle$. We will present estimates of the entropy of urls, queries and IP addresses, taken one at a time. In addition, we will present estimates of the joint entropies of all pairs of these quantities, as well as the three-way joint. From these quantities, we can easily derive estimates of conditional entropies of any combination of these variables (X) given any other combination of these variables (Y) using the rule[3]

$$H(Y|X) = H(X, Y) - H(X)$$

11.2.3 Cross Entropy

It is common practice to split the data into two pieces, one for training and the other for validation. Entropy estimation is fundamentally a prediction task. The task is to use historical logs to estimate search experiences in the future. Splitting up the data into separate training and validation sets tend to produce larger (and more credible) estimates of entropy. Cross entropy[4] can be applied to measure the average number of bits needed to guess the next url in new logs (validation), given the distribution estimated from the historical logs (training).

For example, the cross entropy of url given the query and IP address is

$$H_c(URL|Q, IP) = -\sum_{url,ip,q} p_v(url, ip, q) \log p_t(url|ip, q)$$

where $p_t(url|ip, q)$ is estimated from the training set (historical log data) and $p_v(url, ip, q)$ is estimated from the validation set (new log data). Please note that minimizing this cross entropy is equivalent to maximizing the likelihood of the new log data, given the estimates from the history.

[3] http://en.wikipedia.org/wiki/Conditional_entropy
[4] http://en.wikipedia.org/wiki/Cross_entropy

Based on these evaluation measures, we present a series of experimental results which answers the questions in Section 11.1. How big is the web? How hard is search? How much does personalization help?

11.3 Entropy in Search Logs

11.3.1 How Large is the Web?

How Large is the web? This question refers to the entropy for URL if we know nothing about the context. Entropy estimates for Q, URL and IP addresses are shown in Table 11.1. The entropy estimates are surprisingly small; 22 bits is millions, not billions. A cache of a few million pages will cover much of the demand.

Combination	One Month	1.5 Year
H(Q)	21.14 (25.1)	22.94 (29.2)
H(URL)	22.06 (25.9)	22.44 (29.1)
H(IP)	22.09 (24.6)	22.64 (27.5)

Table 11.1: The search space of the web is surprisingly small; 22 bits of entropy corresponds to a perplexity of millions, not billions.

The numbers in brackets correspond to the entropy if the data is uniformly distributed, or the maximum entropy. The actual estimates are significantly smaller than these upperbounds, and change very slowly with the increase of data.

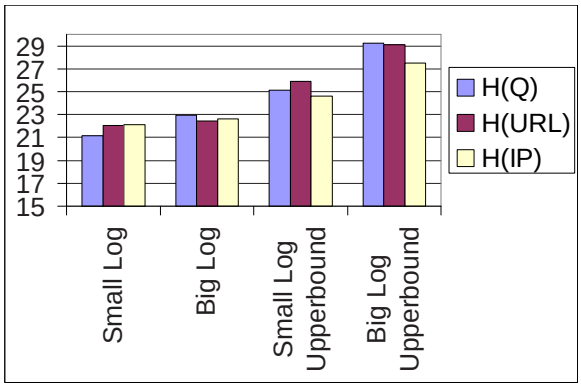


Figure 11.1: Entropy of logs grows much more slowly than its upperbound

Figure 11.1 shows a clearer trend. With the bigger dataset, the maximum entropies (upperbounds) increase significantly (> 3 bits, 1 bit corresponds to a twice larger search space). The actual entropies, however, stay around 22 to 23.

11.3.2 How Hard is Search?

This question refers to the entropy of URL if we know the context of the query, that is, H(URL|Q).

Combination	One Month	1.5 Year
H(Q)	21.14	22.94
H(URL)	22.06	22.44
H(IP)	22.09	22.64
H(Q,URL)	23.88	26.41
H(Q,IP)	26.00	30.41
H(IP,URL)	27.06	31.16
H(Q,URL,IP)	27.17	31.67

Table 11.2: Entropy estimates of all combinations of Q, URL and IP addresses.

Table 11.2 is like Table 11.1, but adds joint entropies for all combinations of Q, URL and IP address. The size of the search space for search can be estimated from Table 11.2. The search task is to guess the URL that the user is looking for from a Query Q. Based on the one month data, that is,

$$\begin{aligned} H(URL|Q) &= H(Q,URL) - H(Q) \\ &= 23.9 - 21.1 = 2.8 \end{aligned}$$

This number becomes *3.5* with the bigger data. In other words, search is doable. A user can often find the url he is looking for somewhere in the top 10 search results. That is reassuring, though not surprising.

We would expect an upper bound around $log_2 10 \approx 3.3$ bits, given the source of the data (click logs). Users tend to click somewhere on the first page of results, or not at all.

11.3.3 How Much Does Personalization Help?

Suppose we give the search engine not only the context of the query, but also the IP address. How much does that help? Using the one month estimates in Table 11.2 above,

$$\begin{aligned} H(URL|Q,IP) &= H(Q,URL,IP) - H(Q,IP) \\ &= 27.2 - 26.0 = 1.2 \end{aligned}$$

In other words, personalization cuts the search space in half. That is a huge opportunity. This entropy becomes *1.3* with the bigger data. Please note that although the joint entropy of the three variables increases a lot, this conditional entropy remains very small.

Why does personalization help? Consider the ambiguous query: MSG. Some users, especially those near

New York City, are looking for the sports arena (Madison Square Garden), whereas other users are looking for the food additive (Mono-sodium Glutamate). The search engine should use the user's history of queries and clicks (when possible) to disambiguate.

11.3.4 How Hard are Query Suggestions?

There are a number of applications that search the space of queries as opposed to the space of answers. For example, a number of query suggestion mechanisms have been proposed suggest as Google Suggests[5] and The Wild Thing [29]. How hard is it to guess the next question, as opposed to guessing the next answer? $H(Q) = 21.1$ bits (22.9 from 1.5 year).

How much does personalization help?

$$\begin{aligned} H(Q|IP) &= H(Q, IP) - H(IP) \\ &= 26 - 22 = 4 \end{aligned}$$

This number becomes *7.8* with the bigger data. In other words, personalization cuts the search space in more than a half. This is a really huge opportunity.

Please note that these entropy estimates can be explained by the notion of context. The basic language units in the task of web search are not English words, but the symbolic token of URLs. According to the relaxed definition in Chapter 3, these URLs are also meaningful language units, which carries rich semantic meanings. If we don't know anything about the context, how many bits does it take to guess the next URL in search? 22 bits. If we know the context of the query, how many bits does it take to guess the URL? Around 3 bits. What about if we know the context of the query and the context of the user? The conditional entropy is cut in half!

In the task of query suggestion, the basic language units are queries, and contexts are users. Once again, if we know the context of the user, we can potentially cut the difficulty of query suggestion in half! The effect of context is huge in the task of web search, which makes contextual text mining in query logs appealing.

The entropy estimates in this section assume that the search log data is seen, and thus correspond to the lower bounds of search difficulty. In reality, when a search engine tries to predict unseen data, the actual entropies (cross entropies) would be higher. Smoothing methods have to be applied on the personalization language models. We will introduce one possible choice in the following section.

[5] http://labs.google.com/suggests

11.4 Personalization with Backoff

The entropy numbers are really exciting. They make a strong case for plausibility, but there are many remaining challenges that need to be addressed including privacy and data sparsity. In fact, Shannon's entropy gives a lower bound of the search difficulty, but does not provide an operational procedure to achieve it. Personalization is very attractive when we have plenty of data, but what if we do not have enough data, or we cannot use much of the data that we have because of privacy concerns? In this section, we introduce one possible operational procedure of approaching this lower bound: personalization with backoff.

The whole procedure presents an instantiation of contextual text mining. In the rest of this section, we show how we select the contexts, construct the contextual language model, and estimate the parameters.

11.4.1 User Modeling with Backoff

If we don't have enough data for a particular user, or we can't use the data we have, we recommend backing off to classes of users. As a proof of concept, we form classes of users based on the first few bytes of the IP address. Even better is to back off based on market segments and collaborative filtering (other users who click similarly). Time and geography can be viewed as surrogate variables for demographics in market segmentation analysis.

The model assumes that users in a class share similar interests. For example, users from the same company are likely to ask similar questions and click on similar answers. Consequently, if we lack adequate historical data for a particular user, we can backoff to a larger class of similar users.

Formally, assume a query q, a user u, and a web document url. Let $\mathcal{U} = \{C_0, C_1, ..., C_{n-1}\}$ be a set of n classes of users. Under personalization with backoff, the probability $p(url|q, u)$ is estimated as a simple linear combination of the class models, for each class that the user is a member of. The weights, λ, can be fit with EM [36]. That is,

$$p(url|q, u) = \sum_{C_i \in \mathcal{U}} \lambda_{u,i} p(url|q, C_i) \tag{11.1}$$

where $\sum_i \lambda_{u,i} = 1$, and $\lambda_{u,i} = 0$ if $u \notin C_i$.

Note that the classes need not form a partition. In particular, we will place IP addresses into a nested hierarchy. Each IP address can be a member of multiple nested classes. Certainly, IP hierarchy is not the only possible choice of the user classes, and perhaps not the best choice either.

This model allows for a wide range of personalization. Two extreme special cases are 0% personalization and 100% personalization. We will refer to 0% personalization as non-personalized, and 100% personalization

as complete personalization.

Non-personalization (or 0% personalization) is the special case where $n = 1$. There is just one super-class of users: $\mathcal{U} = \{C_0\}$. All users are members of this single super-class. In this special case, the model becomes $p(url|q, u) = p(url|q, C_0)$, where C_0 can be dropped.

At the other extreme, 100% personalization, $n = |U|$. Every class contains exactly one user, and every user belongs to exactly one class. In this special case, the model becomes $p(url|q, u) = p(url|q, C_u)$, where $C_u = \{u\}$.

Between these two extreme cases, there is plenty of middle ground, where users are grouped into more than one class, but less than $|U|$. Class assignments are typically determined by variables such as IP addresses, time and geography, and combinations thereof. These variables can be treated as surrogates for demographic variables.

11.4.2 Nested Classes Based on IP Addresses

Users are assigned to 5 nested classes based on their IP address. It is assumed that the prefix of an IP address is a convenient surrogate for some more meaningful variables such as geography. An IP address consists of four sections, each of which is typically encoded with a byte. This representation suggests the following 5 classes:

- IP_4: Users are assigned to classes based on all 4 bytes of the IP address.
- IP_3: Users are assigned to classes based on the 3 most significant bytes of the IP address.
- IP_2: Users are assigned to classes based on the 2 most significant bytes of the IP address.
- IP_1: Users are assigned to classes based on the most significant byte of the IP address.
- IP_0: All users are assigned to a single super-class.

With this construction, every user is assigned to exactly 5 classes. IP_4 and IP_0 are the two extreme cases mentioned above: 100% personalization and 0% personalization, respectively.

We further simplify the model to use just 5 λ's. That is,

$$\begin{aligned} p(url|q,ip) &= \lambda_0 p(url|q,C_{ip,0}) + \\ &\quad \lambda_1 p(url|q,C_{ip,1}) + \\ &\quad \lambda_2 p(url|q,C_{ip,2}) + \\ &\quad \lambda_3 p(url|q,C_{ip,3}) + \\ &\quad \lambda_4 p(url|q,C_{ip,4}) \end{aligned} \tag{11.2}$$

where $C_{ip,k}$ is the class of IP addresses that share the most significant k bytes of ip.

For for example, the IP address, ip =156.111.188.243, belongs to 5 nested classes, namely:

$$\begin{aligned} C_{ip,4} &= \{156.111.188.243\} \\ C_{ip,3} &= \{156.111.188.*\} \\ C_{ip,2} &= \{156.111.*.*\} \\ C_{ip,1} &= \{156.*.*.*\} \\ C_{ip,0} &= \{*.*.*.*\} \end{aligned}$$

There are many ways to fit the λ's. We used the standard Expectation-Maximization(EM) algorithm [36], an iterative procedure which estimates the parameters of the model from the training set, and also finds the λ's that *maximize* the validation set $\mathcal{V}$ given the model. On each iteration, we perform both an estimation (E) step as well as a maximization (M) step with the following two updating formulae:

$$z^{(n+1)}_{\langle q,url,ip\rangle,i} = \frac{\lambda^{(n)}_i p(url|q,C_{ip,i})}{\sum_{k=0}^{4} \lambda^{(n)}_k p(url|q,C_{ip,k})}$$

$$\lambda^{(n+1)}_i = \frac{\sum_{\langle q,url,ip\rangle \in \mathcal{V}} z^{(n+1)}_{\langle q,url,ip\rangle,i} C(\langle q,url,ip\rangle,\mathcal{V})}{\sum_{\langle q,url,ip\rangle \in \mathcal{V}} C(\langle q,url,ip\rangle,\mathcal{V})}$$

where $p(url|q,C_{ip,k})$ denotes probability estimates based on the training set, and $C(\langle q,url,ip\rangle,\mathcal{V})$ denotes counts of triples based on the validation set.

Figure 11.2 shows the resulting estimates of λ. The training set is a month of logs (January 2006). The validation set is a single day of logs (February 1st).

Figure 11.2 shows that a little bit of personalization is better than too much or too little. Note that λ_3 and λ_2 are considerably larger than λ_4, λ_1 and λ_0. Too much personalization suffers from sparse data

whereas too little personalization misses the opportunity of personalization.

Interestingly, we also see that λ_0 is larger than λ_1 and λ_4. We see this because the validation set contains many IP addresses that do not appear in the training set.

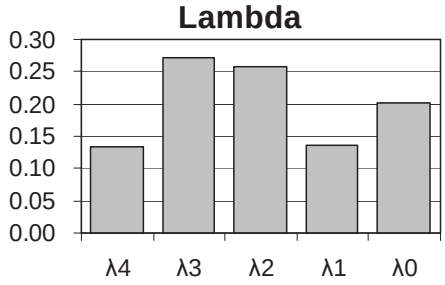


Figure 11.2: A little bit of personalization is better than too much or too little. Note that λ_2 and λ_3 are larger than the other λ's. Too much personalization (λ_4) runs into sparse data, whereas too little (λ_0 and λ_1) misses the opportunity. The EM algorithm assigns more weight to classes of users that share a few bytes of their IP address than to classes that share more (100% personalization) or less (0% personalization).

11.4.3 Connection to the CPLSA Model

It is easy to show that the personalization with backoff models, either the one instantiated with IP addresses (Equation 11.2), or the one generally handles user classes (Equation 11.1), are special cases of the CPLSA model. Of course, we need to relax the basic language units in the CPLSA model, so that the basic observations are URLs instead of words.

What are the contexts here? There is no notion of documents now, where every "document" context in CPLSA now corresponds to a user and the query he issued. Let us assume that there is only one topic in the collection, so there is only one unique (and trivial) topic coverage. But there are many other contexts, defined by the the user classes (C_i) in Equation 11.1 and the IP classes ($C_{ip,i}$) in Equation 11.2. We assume that there is a view for each context, and therefore a query log entry can only select from a few views - corresponding to the user classes that the user belongs to, or the 5 IP classes that the IP address belongs to. For example, in the general model in Equation 11.1, there is a view v_i for every user class C_i; in the IP-based model in Equation 11.2, there is a view $v_{ip,i}$ for every IP class $C_{ip,i}$. We further tie the $\lambda's$ with the view selection probabilities of every "document" (i.e., u, q), so that $p(v_i|u,q) = \lambda_{u,i}$ in Equation 11.1; $p(v_{ip,i}|u,q)$ equals to λ_i in Equation 11.2 if ip is the IP address of the user u, and zero otherwise.

Based on the transformations, we can easily see that both models are special cases of the CPLSA model.

11.4.4 Evaluation of the Backoff Model

How well does this model of personalization perform on future queries? With appropriate smoothing (backoff), personalization should do no harm. Hopefully, personalization improves (reduces) entropy by enough to justify the effort. But no matter what, it should never hurt.

To evaluate the model, we used the logs between February 2, 2006 and February 6, 2006 as a test set, T. We constructed 5 properly nested subsets:

$$T_4 \subseteq T_3 \subseteq T_2 \subseteq T_1 \subseteq T_0 \subseteq T$$

The 5 subsets exclude queries that were not seen in the training set because they could not benefit from this model of personalization. The remainder of the $\langle Q, URL, IP \rangle$ triples in T were assigned to T_0, T_1, T_2, T_3, T_4 based on the most significant k bytes of the IP address. The smallest subset, T_4, contains the triples where all 4 bytes of the IP address were observed in the training set. This set is properly nested within T_3, which contains triples where the first 3 bytes of the of the IP address were observed in training. And so on. Figure 11.3 shows that T_0, T_1, T_2, T_3, T_4 cover between 8.8% and 51.0% of the triples in T.

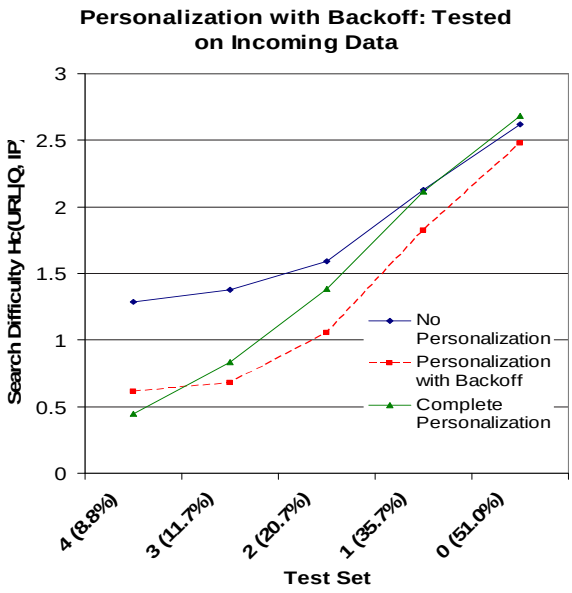


Figure 11.3: In 4 of the 5 test subsets, our proposal, personalization with backoff (dashed lines), has better (lower) cross entropy, $H_c(URL|IP, Q)$, than two baselines: too much personalization (triangles) and too little personalization (solid lines with squares).

Figure 11.3 shows that our proposal, personalization with backoff (dashed lines), does not harm, as we would hope. That is, the dashed line improves (lowers) cross entropy over the "no personalization" baseline,

across all 5 test subsets.

In addition, personalization with backoff beats the "complete personalization" baseline in 4 of the 5 subsets. Obviously, backoff can't beat 100% personalization when you have the relevant data (T_4), but even in that case, backoff isn't much worse.

This section proposed a novel backoff approach to personalization. Backoff is a classic smoothing technique borrowed from language modeling. We show the effectiveness of personalization with backoff using IP addresses. Users are assigned to nested classes, based on the most significant bytes of their IP address. This approach is just one possible way to approach the lower bound of search difficulty estimated with Shannon's entropy. To build a real personalized search engine, this log-based backoff model has to be combined with other features associated with search engines, such as static rank and content relevance. IP address is by no means the only possible surrogate variable for assigning users to classes. The next section will explore some other possibilities.

11.5 The Potential of Other Context Variables

In addition to IP addresses, there are many other context variables that could be used for backing off. The next two subsections will explore day of week and time of day, two variables that have been used to segment telephone traffic into businesses and consumers [33]. Consumers and businesses issue different queries at different times. Different market segments have different needs, and ask different questions at different times. This section shows the potential of incorporating even richer context information into the mining process of personalized search.

Day of Week

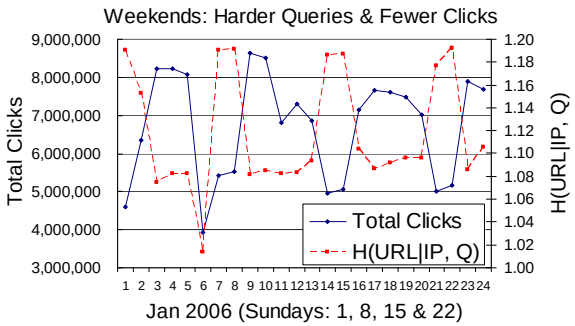


Figure 11.4: Day-of-week patterns could be used to segment the search market into businesses and consumers. Note that click volumes are out of phase with click entropies.

In well-understood mature businesses like telephony, it is common to observe large and important dependencies on day-of-week. Volumes are typically higher on weekdays than weekends. Volumes are especially high on Mondays. Friday afternoon is almost a weekend. The Monday after a long weekend is even bigger than a typical Monday. There are strong interactions between these trends and market segments. Businesses tend to do most of their work on business days, whereas consumers tend to be more active during Prime Time television hours.

Figure 11.4 shows that there are similar day-of-week patterns in the search logs of the first 24 days of January 2006 (where 1/1/06 was a Sunday). As expected, volumes follow the standard pattern, higher during the week than on weekends. Figure 11.4 also shows entropy by day of week, which is more surprising. The queries on business days are easier than on weekends.

This time structure is very repeatable and robust. Figure 11.5 reports cross entropy of personalized search. Each of the first 24 days of January 2006 was used as a training set, and each of the first 6 days in February 2006 was used as a test set. All pairs of a testing and a training set are scored by cross entropy: $H_c(URL|IP, Q)$. Some of the pairs used a weekend in training and some didn't. Similarly, some of the pairs used a weekend in testing and some didn't.

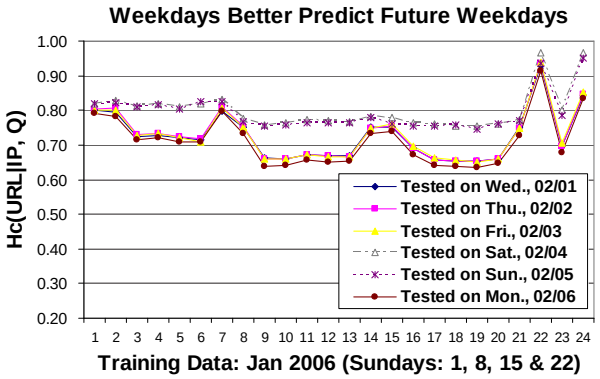


Figure 11.5: Weekends are harder (more entropy) than business days. Cross entropy peaks both when the weekend is used for training and/or testing.

Weekends are harder (larger entropy) than weekdays. Figure 11.5 shows six lines, one for each of the 6 test days. Note that the solid lines (business days) are consistently below the dashed lines (weekends). There are 24 points along the x-axis in Figure 11.5, one for each of the 24 training days. All curves peak on weekends. Weekends are harder, both when used for training as well as testing. From the solid lines, we also learn that future weekdays are better predicted using previous weekdays than using previous weekends.

This analysis suggests that it is potentially beneficial to include the market segmentation of weekdays/weekends along with IP addresses in personalized search, and treat them accordingly.

Time of Day

Figure 11.6 shows query volumes and entropies, $H(Q|IP)$ by hour for the first 15 days of January 2006. There are clear hour of day effects, especially on weekdays.

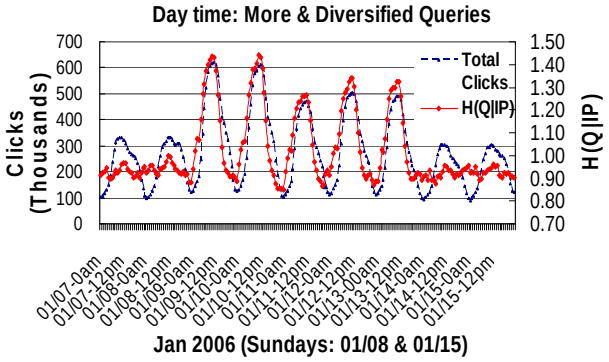


Figure 11.6: Query volumes and entropies show a clear dependence on hour of day, at least for weekdays.

Volumes follow the expected pattern, with more queries during the day and fewer at night. Yet again, entropy is a surprise. Recall that volumes and entropies were out of phase with one another in Figure 11.4. This time, they are in phase with one another. It appears that queries are different from clicks.

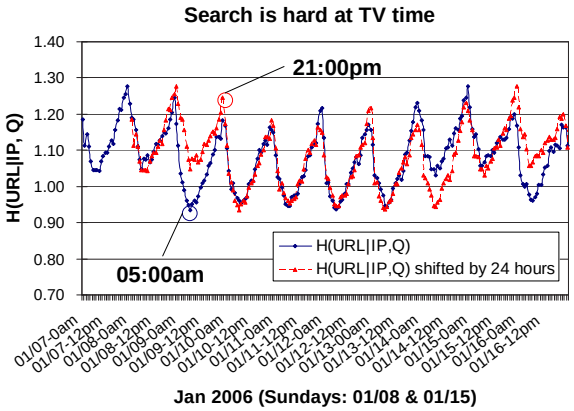


Figure 11.7: Search is simpler at work hours and more difficult at television hours

Figure 11.7 shows entropies of clicks, $H(URL|IP,Q)$, by hour from January 7, 2006 to January 16, 2006. There is a strong hourly time structure. Entropy peaks during prime time TV hours. The valleys are very early in the morning.

The dashed line highlights the hourly time structure. The dash line is the solid line shifted right by 24 hours. An autocorrelation analysis would compare these two lines, showing that there is a strong periodicity with a lag of 24 hours, not surprisingly. The plot makes it clear that the daily periodicity is stronger on weekdays than weekends, which again, is not unexpected.

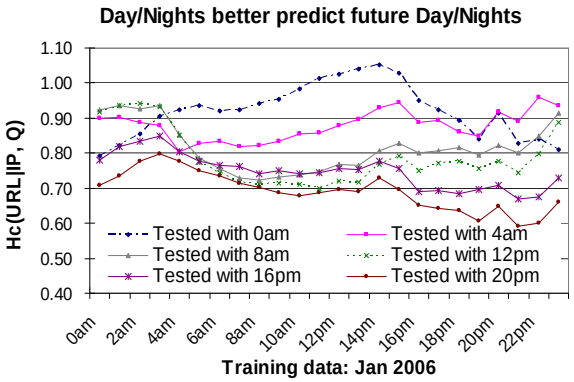


Figure 11.8: Cross validation of the predictive ability of hours in a day

To test whether a market segmentation with time of day could help the personalization of future search activity, we cross validate the search difficulty with logs during different hours of day as training and testing datasets. Specifically, we partition the search logs in January 2006 into 24 training sets, each corresponding to one hour in the day, and select six hours' search logs on February 1, 2006 as testing sets. The results of H_c(URL|IP, Q) are plotted in Figure 11.8. From the plots, we see that the search history during different hours of a day shows different predictive ability over a testing set of different hours. From the two dashed lines, it is easy to see that search in the day time can be better predicted by history of the day time, and nights are better predicted by nights. When the time of the training set is closer to the time of the testing set, the cross entropy becomes lower. When the time of the training set departs from the time of the testing set, H_c(URL|IP, Q) becomes larger. This suggests that the best training data set to personalize future search activity at a given time of day is the search history at the closest time period of a day.

In this analysis, it is clear that segmenting the search business with the time of day is potentially beneficial on top of IP address and day of week.

Query Types

We already presented that market segmentation with geographic variables such as IP addresses, and time variables such as day/week and hour/day are beneficial for personalization. All these variables are metadata variables in search. How about those core variables in search? Is it also beneficial to differentiate the core variable in search, the query?

Figure 11.9 and Figure 11.10 present the day-of-week frequency patterns for two groups of queries. The first group includes three query strings "yahoo," "mapquest," and "cnn," and the second group includes "sex," "mp3," and "movies." From Figure 11.9, we see that the frequency of the first group of queries, which we shall call business queries, has clear day-of-week patterns. There are significantly more business

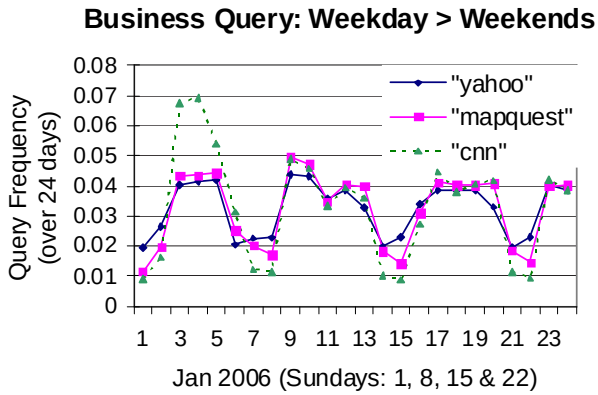


Figure 11.9: Business queries are issued on business days.

queries on weekdays than weekends. The second group of queries, which we shall call consumer queries, however, does not show clear day-week patterns. As in Figure 11.10, the frequency of consumer queries on weekends is comparable, sometimes even higher than their frequency on weekdays. It is interesting (but not unexpected), that the query "movies" is asked most frequently on Fridays.

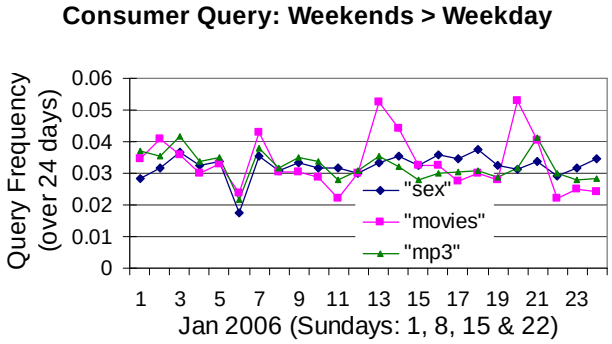


Figure 11.10: Unlike business queries, consumer queries do not select for business days.

Figure 11.11 presents the comparison of the time-of-day patterns of the frequency of two queries, "yahoo" and "sex". It is clear that both queries have clear time-of-day patterns. However, the frequency of these two queries gets highest and lowest at different hours.

This analysis shows that besides metadata variables such as geography and time, the search business can also potentially benefit from segmenting the search space with the type of core search variables, such as the queries.

In general, users ask more questions and simpler questions during business days and business hours. It isn't clear why this is so, but we might hypothesize that businesses are more business-like, more likely to ask direct questions that have direct answers, like navigational queries. "cnn" is an example of a navigational query. The answer to the "cnn" query is simply: "www.cnn.com." In contrast, consumers ask less direct

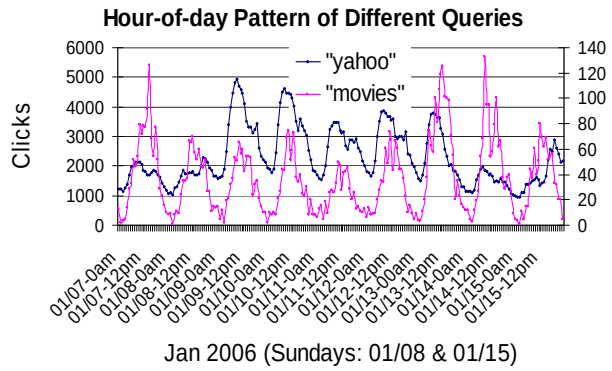


Figure 11.11: Different types of queries have different hour-day patterns

questions. They may be seeking employment, health, wealth, happiness, entertainment, etc. They may be shopping or just browsing, with no particular place to go, and lots of time on their hands. In extreme cases, one can even find Eliza-like[6] queries in the logs.

11.6 Related Work

This chapter draws connections across a wide range of fields including: Information Theory, backoff smoothing of language models, query suggestions, personalization and marketing. There is a huge body of work in each of these areas, though relatively little work that connects all of them (or even many of the pairs).

Entropy has a long history dating back to Shannon, and perhaps, earlier. See [47] for an excellent retrospective on Shannon's life, work and impact. Entropy has been applied to many data sources, though there is still plenty of room for novel applications such as web logs.

There is a considerable body of work on estimating the size of the web including: [85, 7, 86, 87, 37, 54]. These references attempt to estimate supply (the size of the reachability graph of links from one url to the next), as opposed to demand (clicks). By taking demand into account, we can come up with much more feasible estimates, millions not billions, suggesting that a small cache of a few million pages could capture much of the value.

Entropy analysis is once again well accepted in Computational Linguistics. Back in the late 1940s and early 1950s, Shannon's Information Theory was having a dramatic impact on a wide range of fields from Engineering to Psychology and more. Shannon published his remarkable estimate of the size of English language in 1951 [154]. Chomsky's *Syntactic Structures* [25] came out shortly thereafter in 1957, arguing that language was unbounded (infinite) and that ngram approximations (such as Shannon's) do not come closer and closer to the truth. Chomsky's work dominated much of the thinking in Computational Linguistics

[6] http://en.wikipedia.org/wiki/ELIZA

over the next few decades, but Information Theory regained popularity in Computational Linguistics in the 1990s with the successes of trigram language models in speech recognition [28]. The speech application motivated researchers to think about smoothing methods such as backoff: [68, 26, 22, 183].

Query suggestions [29, 66, 67, 174] and personalization [65, 62, 163, 166, 155, 156] are somewhat related topics, though the connection between those two topics and backoff smoothing of language models is novel. Many personalized search techniques have been proposed, both server-side [166][7] and client-side [156, 164], as well as with long term query history [166, 164] and short term implicit feedback [155, 143]. Much of this work takes advantage of search engine query logs. This work tends to be focused more on methods of improving user experience, and less on sizing challenges and opportunities.

Market segmentation (and demographics) come from a completely different tradition than Language Modeling and backoff. Marketing is relatively more central to this conference than Information Theory and Language Modeling. Pregibon and Cortes [33], for example, were concerned with marketing applications in telephony. Marketing was eager to find ways to segment customers based on usage (demand). Pregibon and Cortes found that businesses and consumers make calls at different times for different purposes. Marketing could take advantage of such insights to target offers more appropriately since businesses and consumers respond differently to different offers such as various pricing plans and discounts. The connection between, marketing, a well-established KDD application, and Language Modeling is novel.

11.7 Summary

In this chapter, we presented a particular instantiation contextual text mining, which leads to an effective solution for a real world text mining tasks - personalized web search. We proposed a model called personalization with backoff, which can be shown as a special case of the general mixture model to handle a type of explicit contexts - users and groups of users in web search. We also showed how entropy can be used to address a number of fundamental questions in web search, and to evaluate a contextual language model. Entropy was estimated from search logs, a sequence of triples: $\langle Q, URL, IP \rangle$, indicating that a click was observed from a particular URL and a particular IP address in response to a particular query q.

How hard is search? It takes around 22 bits to guess the next url (or the next query or the next user). 22 bits is millions, not billions. When we give the search engine the query, we cut the 22 bits down to around 3 bits. What is the opportunity for personalization? Personalization cuts the search space in half (from 3 down to less than 1.5 bits). That is a huge opportunity. A personalized cache is an even bigger threat to the cluster in the cloud than a cache without personalization.

[7] See also www.google.com/psearch.

Shannon's entropy provides a novel way to think about sizing challenge and opportunity in search business. It gives the lower bound of the difficulty of personalized search but not an operational procedure to approach this lower bound. In reality, when we do not have data for a particular user, a smoothed version of the language model $P(URL|Q,U)$ has to be applied. While different smoothing methods lead to different personalization approaches, we introduce one of those choices: personalization with backoff.

Personalization with backoff is more effective than personalization without backoff. As a proof of concept, we discussed backing off to classes of users based on IP addresses. A little bit of personalization was found to be better than too much or too little. This exploration illustrates the application of contextual text mining in the task of web mining. The proposed models of personalization with backoff are once again special cases of the CPLSA model. Contextual text mining is powerful. We can see that with such a straightforward instantiation of the CPLSA model, we can already facilitate a difficult problem like personalized search in an effective way. The data we explore is very large (as large as the 18-month search log of a commercial search engine). The instantiation of contextual text mining works effectively on such a large scale dataset.

Rather than backing off based on prefixes of IP addresses, it would be better to back off based on market segmentation (demographics), or other context variables in general. Different segments have different needs (ask different questions at different times) and different values (willingness to pay and advertising opportunities). Businesses and consumers ask different questions at different times. We showed that query volumes and search difficulty (entropy) vary by time of day and day of week. There is a huge potential to incorporate other context variables into personalization with backoff.

Chapter 12

Conclusions

In this thesis, we presented contextual text mining, a new paradigm of text mining which treats context information as the "first class citizen." In this chapter, we come to the conclusions of this thesis and discuss the future directions of exploration in contextual text mining.

12.1 Summary

Text is usually associated with all kinds of context information. Some contextual information is explicit, similar to the metadata in a database, such as the time and the location where a blog article is written, and the author(s) of a biomedical publication. There is also implicit context, such as the positive or negative sentiment that a user had when she wrote a product review. Contexts could be dependent to each other, and thus form a complex structure, such as social networks.

Context information is very important in understanding text data because it significantly affects the choices of words, topics, and other user behaviors in text. In a way, the generative process of text is like the selection process of pieces in a Jigsaw puzzle, which depends on a variety of contexts, such as shape, color, etc. Understanding the context is crucial in many text mining tasks. Many applications require analysis of patterns of text over different contexts. For instance, click patterns of different users in search logs can help enhance the quality of a commercial search engine by re-organizing the search results based on particular users. The evolutionary patterns of topics in scientific literature would help researchers better digest the trend of research and envision hot research topics. The distribution of opinions on a social network is very valuable to researchers in sociology and business. Context plays an important role in text mining to help understand the meanings of language units and characterize the variation of these semantics under different situations.

This thesis provides the first formal and systematic study of context in text mining. Instead of narrow definitions of context, such as linguistic context and social context, we focus on a much more general definition of context - context of situation. In this thesis, we innovatively defined the key concepts in contextual text

mining, formulated the general tasks of contextual text mining, summarized the basic principles of contextual text mining, and introduced a unified theoretical framework with which we can solve many specific problems like those mentioned above. The general framework not only includes the construction of a contextual language model that explains the generative process of text based on context, a component to constrain and regularize the contextual language model with prior knowledge and important assumptions, a component to extract a variety of contextual patterns, but also a phase to postprocess the basic contextual patterns in order to extract refined patterns or facilitate various learning tasks. Indeed, many text mining problems with context involved can be shown as instantiations of the general framework, and thus can be solved by using the general methodology along with this framework.

To bridge the gap between this general conceptual framework and the real applications, we further instantiated the conceptual framework and proposed a functional framework of contextual text mining with topics involved as one type of contexts. This functional framework, called contextual topic analysis, is not only general enough to handle different types of contexts and extract various types of contextual patterns, but also specific enough to be directly applied to a certain real world text mining task.

Such a functional framework contains a general contextual topic model, which is extended from the probabilistic latent semantic indexing by incorporating context variables in a general way. With the proposed contextual topic model, called the CPLSA model, one can easily extract the latent views and coverages of topics in the text collection, and compare them across different contexts. According to the specific needs of real tasks, the CPLSA model can be instantiated into many special cases, including the PLSA model, a Fixed-View CPLSA model, a Fixed-Coverage CPLSA model, a Topic-Sentiment Mixture model, a NetPLSA model, and a personalization with backoff model for web search, etc. These special cases, including the general CPLSA model itself, have already been applied in a wide range of real applications of contextual topic analysis and performance effectively.

There are two general components to further refine the contextual topic model in this framework. One component allows us to add priors to the contextual topic model through which we can incorporate important prior knowledge about the context and the generative process. Such a prior can come from the user's knowledge about the contexts, the domain knowledge of the task, and knowledge learned from a training data. By doing this, we provide a novel way to bring users into the loop, so that a user can guide and steer the text mining system with her prior knowledge and preference. This component is powerful, especially in handling implicit contexts.

Another general component to refine the contextual topic model is to regularize the model parameters based on the dependency among contexts. This component regularizes the likelihood of the data to be

generated by the contextual topic model with objectives defined based on the complex structure of contexts. By bridging probabilistic generative models with graph-based regularization, this component provides a general way to incorporate important assumptions and principles about the dependency, similarity, and correlation between contexts into the mining process. This component is powerful in handling complex contexts, such as social networks.

The inference with the contextual language model provides us a rich set of context models and contextual patterns that can be used to characterize the text data. Our exploration doesn't stop here. Instead, we advanced beyond this step, by further postprocessing the context models and contextual patterns. This postprocessing procedure lead to refined contextual patterns such as the evolutionary theme patterns, which can reveal deeper and more interesting knowledge in the data. We proposed an innovative method to generate meaningful labels for contextual patterns such as topic models. This method successfully address the problem of how to improve the interpretability of mining results, which is a fundamental problem in data mining in general.

Another contribution of this thesis is the illustration of a wide range of real applications of contextual text mining. We show the effectiveness of our proposed framework, contextual topic models, and other contextual text mining techniques in the "context" of real world text mining tasks, including temporal text mining (Chapter 8), spatiotemporal topic analysis, temporal author-topic analysis, event-impact analysis (Chapter 5), faceted opinion summarization, sentiment dynamics tracking (Chapter 6), topical community extraction, and topic map construction (Chapter 7). These special cases of contextual text mining covered a handful set of different contexts, including explicit contexts like time, geographic location, and authorship, implicit contexts like topics and sentiments, as well as complex contexts such as social networks.

Contextual text mining is powerful, not only in its own instantiations but also in accomplishing other text information management tasks. We applied specific contextual text mining techniques in two well explored tasks - text retrieval and web search. A special case of the NetPLSA model (contextual topic model with network regularization) can help smoothing both document language models and query language models using a document graph or a word graph. This application of contextual text mining leads to several novel smoothing methods which consistently outperform the state-of-the-art smoothing methods for language models in information retrieval. In addition, we applied a special case of the CPLSA model in search engine logs, with the consideration of search users as the context. Based on entropy analysis in search logs, we show that the difficulty of web search can be potentially cut down to half with the special contextual language model, personalization with backoff. The applications of contextual text mining in these two tasks also provide a quantitative proof of the effectiveness of contextual text mining on large scale text collections.

The contextual text mining techniques we developed have been applied in a wide range of tasks beyond this thesis, such as generating semi-structured gene summaries from biomedical literature [94], mining multi-faceted overviews from text [95], summarizing the impact of scientific literature [115], tagging web objects [117], and generating personalized query suggestions [118].

12.2 Future Directions

Contextual text mining is a new paradigm of text mining which opens a lot of new possibilities for future research of text mining. The framework we proposed, the techniques we developed, and the models we constructed are just an initial step in understanding the correlation between text and context in text mining. There are many interesting future research directions following the discussion in this thesis, including:

Beyond Bag-of-Words

In this thesis, we follow the existing work and make the assumption to represent text documents with a bag of words. One interesting direction is to relax this assumption, and explore more flexible levels of language units and the dependency among language units. Smoothing language models using word graph in Chapter 10 is along this line, where the dependency structure among words is utilized to regularize the contextual language model. Beyond this, there are still many interesting research topics. How to model the generative process of phrases? How to model the proximity of words? How to model the local linguistic context in topic models? All these questions lead to intriguing future work.

Contextual Extension to Other Generative Models

In Chapter 4, we present an extension of PLSA as a general contextual topic model. Such a contextual extension can be expected to apply on other generative models, such as the Latent Dirichlet Allocation (LDA), the Hidden Markov Models (HMM), and Poisson mixtures. It is appealing to explore this in future research work. The contextual extension of those models are expected to both leverage the power of contextual analysis and inherit the advantages of these models. For example, LDA can alleviate the overfitting problem of PLSA, HMM captures the sequential dependency among words, and Poisson language models provide more flexibility than multinomial models [109].

Contextual Text Mining with Discriminative Models

In this thesis, we follow the generative view of text mining, by modeling and analyzing text data with probabilistic generative models. The alternative approach to generative modeling is discriminative modeling (e.g., supporting vector machine, conditional random field, etc), which has been proved effective in many text mining tasks such as text classification, labeling, and learning to rank. An interesting future direction is to study the impacts of contexts in the setting of discriminative learning. How can we model context information in discriminative models? Are there more effective ways of using context information, other than treating it as additional features in the representation of text? These questions can be answered by future research on context modeling in discriminative models.

Automatic Context Selection

For a text collection with rich context information, there can be many possible contexts. Usually, the text mining task itself can tell us what contexts are useful and what are not. For example, to extract topic life cycles, apparently we should choose the time context and not the location. In this thesis, we make the assumption that we already know the target contexts. However, for other tasks like contextualized search, it is difficult even to figure out what context information is useful. In such a case, we need a method to automatically select, or suggest contexts for us. The estimation of entropy is a primitive exploration on this direction. This problem also relates to the feature selection problem for machine learning.

User-centered Contextual Text Mining

Another important direction for contextual text mining is to get a real user involved in the mining process. A user-centered contextual text mining system should allow a user to flexibly select the data and context, and adjust the contexts and models based on the feedback of users. For example, a user should be able to drill contextual patterns down to a finer granularity; she should also be able to navigate from one context to a related context. A contextual language model should also be able to adapt prior knowledge input from the users and learn from the history of user behaviors.

Domain Specific Applications

Contexts are multifarious. There are a wide variety of contexts in text, which provide numerous opportunities for specific applications of contextual text mining. One important direction is to explore the effect of context to text data generated in specific domains, such as content of social communications, scientific literature,

and user-generated data in Web2.0 applications. Applications in these domains focus on specific types of context, such as social context and Psychological context. There is rich prior knowledge and understanding of these contexts' influence over the content in these domains. How to model the specific types of context to support specific applications in these domains? How to incorporate specific prior knowledge about context into contextual text mining? How to leverage contextual text mining to answer research questions in these domains? All these questions are likely to lead to novel interdisciplinary research directions.

Novel Information Systems

Finally, it is exciting to tune the techniques for particular applications and to build real information systems of contextual text mining. Such systems should satisfy the real needs of users. They should also be able to process large scale real data effectively and efficiently. Testing the power of contextual text mining with end-user applications and real systems is the best way to evaluate the performance of the contextual text mining techniques.

To summarize, with the interaction of text and context, there are numerous opportunities in knowledge discovery from text data.

References

[1] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study: Final report. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

[2] J. Allan, R. Gupta, and V. Khandelwal. Temporal summaries of news topics. In *Proceedings of SIGIR '01*, pages 10–18, 2001.

[3] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of KDD '06*, pages 44–54, 2006.

[4] S. Banerjee and T. Pedersen. The design, implementation, and use of the ngram statistics package. pages 370–381, 2003.

[5] R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *Proceedings of ICML*, pages 41–48, 2005.

[6] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, 2003.

[7] K. Bharat and A. Broder. A technique for measuring the relative size and overlap of public web search engines. In *Proceedings of the seventh international conference on World Wide Web 7*, pages 379–388, 1998.

[8] D. Blei and J. Lafferty. Correlated topic models. In *NIPS '05: Advances in Neural Information Processing Systems 18*, 2005.

[9] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120, 2006.

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[11] A. Bookstein and D. R. Swanson. Probabilistic models for automatic indexing. *Probabilistic Models for Automatic Indexing*, 25(5):312–318, 1974.

[12] S. Borgatti, M. Everett, and L. Freeman. Ucinet for windows: Software for social network analysis. *Harvard: Analytic Technologies*, 2002.

[13] S. Boykin and A. Merlino. Machine learning of event segmentation for news on demand. *Commun. ACM*, 43(2):35–41, 2000.

[14] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998.

[15] P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Comput. Linguist.*, 16(2):79–85, 1990.

[16] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR '98*, pages 335–336, 1998.

[17] P. K. Chan, M. D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. In *Proceedings of the 30th international conference on Design automation*, pages 749–754, 1993.

[18] E. Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139, 2000.

[19] C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman and Hall, 1996.

[20] C. C. Chen, M. C. Chen, and M.-S. Chen. Liped: Hmm-based life profiles for adaptive event detection. In *Proceeding of KDD '05*, pages 556–561, 2005.

[21] J. Chen, J. Yan, B. Zhang, Q. Yang, and Z. Chen. Diverse topic phrase extraction through latent semantic analysis. In *Proceedings of ICDM '06*, pages 834–838, 2006.

[22] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318, 1996.

[23] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University, 1998.

[24] Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of HLT-EMNLP 2005*, 2005.

[25] N. Chomsky. *Syntactic Structures*. The Hague/Paris: Mouton, 1957.

[26] K. Church and W. Gale. A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of english bigrams. *Computer Speech and Language*, 5(1):19–54, 1991.

[27] K. Church and W. Gale. Poisson mixtures. *Nat. Lang. Eng.*, 1(2):163–190, 1995.

[28] K. Church and R. Mercer. Introduction to the special issue on computational linguistics using large corpora. *Computational Linguistics*, 19(1):1–24, 1993.

[29] K. Church and B. Thiesson. The wild thing! In *Proceedings of the ACL 2005*, pages 93–96, 2005.

[30] K. W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on Applied natural language processing*, pages 136–143, 1988.

[31] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, 1990.

[32] D. A. Cohn and T. Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *NIPS*, 2000.

[33] C. Cortes and D. Pregibon. Signature-based methods for data streams. *Data Min. Knowl. Discov.*, 5(3):167–182, 2001.

[34] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[35] W. B. Croft and J. Lafferty, editors. *Language Modeling and Information Retrieval*. Kluwer Academic Publishers, 2003.

[36] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statist. Soc. B*, 39:1–38, 1977.

[37] D. Dhyani, W. K. Ng, and S. S. Bhowmick. A survey of web metrics. *ACM Comput. Surv.*, 34(4):469–503, 2002.

[38] F. Diaz. Regularizing ad hoc retrieval scores. In *Proceedings of CIKM' 05*, pages 672–679, 2005.

[39] K. Eguchi and V. Lavrenko. Sentiment retrieval using generative models. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 345–354, July 2006.

[40] C. Engstrom. Topic dependence in sentiment classification. masters thesis. university of cambridge. 2004.

[41] U. Fayyad, D. Haussler, and P. Stolorz. Mining scientific data. *Commun. ACM*, 39(11):51–57, 1996.

[42] R. Feldman and I. Dagan. Knowledge discovery in textual databases (kdt). In *KDD*, pages 112–117, 1995.

[43] J. Firth. Ethnographic analysis and language with reference to malinowskis views. *Man and Culture: An Evaluation of the Work of Bronislaw Malinowski.*, 1957.

[44] J. R. Firth. A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis*, pages 1–32, 1957.

[45] W. Francis and H. Kucera. Frequency analysis of english usage: Lexicon and grammar. *Houghton Mifflin Company*, 1982.

[46] W. Gale. Good-turing smoothing without tears. *Journal of Quantitative Linguistics*, 2, 1994.

[47] R. Gallager. Claude E. Shannon: A retrospective on his life, work, and impact. *IEEE Transactions on Information Theory*, 47(7):2681–2695, 2001.

[48] D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Comput. Linguist.*, 28(3):245–288, 2002.

[49] K. E. Gill. Blogging, rss and the information landscape: A look at online news. In *WWW 2005 Workshop on the Weblogging Ecosystem*, 2005.

[50] N. Glance, M. Hurst, and T. Tornkiyo. Blogpulse: Automated trend discovery for weblogs. In *WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*, 2004.

[51] T. L. Griffiths and M. Steyvers. Fiding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl.1):5228–5235, 2004.

[52] D. Gruhl, R. Guha, R. Kumar, J. Novak, and A. Tomkins. The predictive power of online chatter. In *Proceedings of KDD '05*, pages 78–87, 2005.

[53] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *Proceedings of the 13th International Conference on World Wide Web*, pages 491–501, 2004.

[54] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 902–903, 2005.

[55] J. Hammerton, M. Osborne, S. Armstrong, and W. Daelemans. Introduction to special issue on machine learning approaches to shallow parsing. *J. Mach. Learn. Res.*, 2:551–558, 2002.

[56] M. A. Hearst. Untangling text data mining. In *Proceedings of the 37th conference on Association for Computational Linguistics (ACL 1999)*, pages 3–10, 1999.

[57] M. A. Hearst. Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61, 2006.

[58] D. Hiemstra and W. Kraaij. Twenty-one at TREC-7: Ad-hoc and cross-language track. In *Proceedings of TREC 7*, pages 227–238, 1998.

[59] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of '99.*

[60] T. Hofmann. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *IJCAI' 99*, pages 682–687, 1999.

[61] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of SIGIR '99*, pages 50–57, 1999.

[62] G. Jeh and J. Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279, 2003.

[63] F. Jelinek. Self-organized language modeling for speech recognition. *Readings in speech recognition*, 1990.

[64] R. Jin and A. G. Hauptmann. A new probabilistic model for title generation. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, 2002.

[65] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

[66] R. Jones and D. C. Fain. Query word deletion prediction. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 435–436, 2003.

[67] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, pages 387–396, 2006.

[68] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speeech and Signal Processing*, 35(3):400–401, 1987.

[69] P. J. Kaufman, Leonard; Rousseeuw. *Finding groups in data. an introduction to cluster analysis.* Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics. Wiley. New York., 1990.

[70] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 102–111, 2002.

[71] J. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170, 2000.

[72] J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of KDD '02*, pages 91–101, 2002.

[73] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632.

[74] A. Kontostathis, L. Galitsky, W. M. Pottenger, S. Roy, and D. J. Phelps. A survey of emerging trend detection in textual data mining. *Survey of Text Mining*, pages 185–224, 2003.

[75] R. Krovetz. Viewing morphology as an inference process. In *Proceedings of SIGIR '93*, pages 191–202, 1993.

[76] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *Proceedings of the 12th International Conference on World Wide Web*, pages 568–576, 2003.

[77] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. Structure and evolution of blogspace. *Commun. ACM*, 47(12):35–39, 2004.

[78] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber-communities. In *Proceeding of the eighth international conference on World Wide Web*, pages 1481–1493, 1999.

[79] O. Kurland and L. Lee. Corpus structure, language models, and ad hoc information retrieval. In *Proceedings of SIGIR' 04*, pages 194–201.

[80] O. Kurland and L. Lee. Pagerank without hyperlinks: structural re-ranking using links induced by language models. In *Proceedings of SIGIR '05*, pages 306–313.

[81] O. Kurland and L. Lee. Respect my authority!: Hits without hyperlinks, utilizing cluster-based language models. In *Proceedings of SIGIR '06*, pages 83–90.

[82] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'01*, pages 111–119.

[83] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 111–119, 2001.

[84] V. Lavrenko and B. Croft. Relevance-based language models. In *Proceedings of SIGIR'01*, pages 120–127.

[85] S. Lawrence and C. L. Giles. Searching the World Wide Web. *Science*, 280(5360):98–100, 1998.

[86] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Nature*, 400(6740):107–109, 1999.

[87] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Intelligence*, 11(1):32–39, 2000.

[88] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceeding of KDD '05*, pages 177–187, 2005.

[89] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst. Cascading behavior in large blog graphs. In *Proceeding of SDM' 07*, 2007.

[90] W. Li and A. McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of ICML*, pages 577–584, 2006.

[91] Z. Li, B. Wang, M. Li, and W.-Y. Ma. A probabilistic model for retrospective news event detection. In *Proceedings of SIGIR '05*, pages 106–113, 2005.

[92] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774, 1998.

[93] D. Lin and P. Pantel. Concept discovery from text. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, 2002.

[94] X. Ling, J. Jiang, X. He, Q. Mei, C. Zhai, and B. R. Schatz. Generating semi-structured gene summaries from biomedical literature. *Information Processing and Management (IPM)*, 2007.

[95] X. Ling, Q. Mei, C. Zhai, and B. Schatz. Mining multi-faceted overviews of arbitrary topics in a text collection. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–505, 2008.

[96] B. Liu, M. Hu, and J. Cheng. Opinion observer: analyzing and comparing opinions on the web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 342–351, 2005.

[97] X. Liu and W. B. Croft. Cluster-based retrieval using language models. In *Proceedings of SIGIR' 04*.

[98] B. Long, Z. M. Zhang, X. Wu, and P. S. Yu. Spectral clustering for multi-type relational data. In *ICML*, pages 585–592, 2006.

[99] J. Ma and S. Perkins. Online novelty detection on temporal sequences. In *Proceedings of KDD '03*, pages 613–618, 2003.

[100] R. E. Madsen, D. Kauchak, and C. Elkan. Modeling word burstiness using the dirichlet distribution. In *Proceedings of the 22nd international conference on Machine learning*, pages 545–552, 2005.

[101] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. W. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of KDD '04*, pages 236–245, 2004.

[102] C. D. Manning and H. Schtze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[103] E. L. Margulis. Modelling documents with multiple poisson distributions. *Inf. Process. Manage.*, 29(2):215–227, 1993.

[104] A. McCallum, A. Corrada-Emmanuel, and X. Wang. Topic and role discovery in social networks. In *IJCAI*, pages 786–791, 2005.

[105] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification, 1998.

[106] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley, 1997.

[107] Q. Mei, D. Cai, D. Zhang, and C. Zhai. Topic modeling with network regularization. In *Proceeding of WWW'08*, pages 101–100, 2008.

[108] Q. Mei and K. Church. Entropy from search logs: How hard is search? with personalization? with backoff? In *Proceeding of WSDM'08*, pages 45–54, 2008.

[109] Q. Mei, H. Fang, and C. Zhai. A study of poisson query generation model for information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 319–326, 2007.

[110] Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *Proceedings of WWW '07*, 2007.

[111] Q. Mei, C. Liu, H. Su, and C. Zhai. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *Proceedings of WWW '06*, pages 533–542, 2006.

[112] Q. Mei, X. Shen, and C. Zhai. Automatic modeling of multinomial topic models. In *Proceeding of KDD'07*, pages 490–499, 2007.

[113] Q. Mei and C. Zhai. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *Proceeding of KDD'05*, pages 198–207, 2005.

[114] Q. Mei and C. Zhai. A mixture model for contextual text mining. In *Proceedings of KDD '06*, pages 649–655, 2006.

[115] Q. Mei and C. Zhai. Generating impact-based summaries for scientific literature. In *Proceedings of ACL-08*, pages 816C–824, 2008.

[116] Q. Mei, D. Zhang, and C. Zhai. A general framework for smoothing language models on a graph structure. In *Proceeding of SIGIR'08*, 2008.

[117] Q. Mei and Y. Zhang. Automatic web tagging and person tagging using language models. In *ADMA*, pages 741–748, 2008.

[118] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 469–478, 2008.

[119] R. Mihalcea and D. R. Radev, editors. *Textgraphs: Graph-based methods for NLP*, 2006.

[120] D. H. Miller, T. Leek, and R. Schwartz. A hidden Markov model information retrieval system. In *Proceedings of SIGIR 1999*, pages 214–221, 1999.

[121] T. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of UAI' 01*, pages 352–359, 2001.

[122] G. Mishne and M. de Rijke. MoodViews: Tools for blog mood analysis. In *AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW 2006)*, pages 153–154, 2006.

[123] G. Mishne and N. Glance. Predicting movie sales from blogger sentiment. In *AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW 2006)*, 2006.

[124] S. Morinaga and K. Yamanishi. Tracking dynamics of topic trends using a finite mixture model. In *Proceedings of KDD '04*, pages 811–816, 2004.

[125] F. Mosteller and D. Wallace. Inference and disputed authorship: The federalist. *Addison-Wesley, Massachusetts, series in behavioral science:quantitative methods edition*, 1964.

[126] R. Nallapati, A. Feng, F. Peng, and J. Allan. Event threading within news topics. In *Proceedings of the Thirteenth ACM conference on Information and knowledge management*, pages 446–453, 2004.

[127] R. M. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. pages 355–368, 1999.

[128] D. B. Neill, A. W. Moore, M. Sabhnani, and K. Daniel. Detection of emerging space-time clusters. In *Proceedings of KDD '05*, pages 218–227, 2005.

[129] D. Newman, C. Chemudugunta, and P. Smyth. Statistical entity-topic models. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 680–686, 2006.

[130] Opinmind. http://www.opinmind.com.

[131] M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Names and similarities on the web: fact extraction in the fast lane. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 809–816, 2006.

[132] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278, 2004.

[133] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, pages 115–124, 2005.

[134] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.

[135] P. Pantel and D. Lin. Discovering word senses from text. In *Proceedings of KDD '02*, pages 613–619, 2002.

[136] P. Pantel and D. Lin. Automatically discovering word senses. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 21–22, 2003.

[137] J. Perkio, W. Buntine, and S. Perttu. Exploring independent trends in a topic-based search engine. In *Proceedings of WI'04*, pages 664–668, 2004.

[138] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, 1998.

[139] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of SIGIR 1998*, pages 275–281, 1998.

[140] T. Qin, T.-Y. Liu, X.-D. Zhang, Z. Chen, and W.-Y. Ma. A study of relevance propagation for web search. In *Proceedings of SIGIR 2005*, pages 408–415, 2005.

[141] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–285, Feb. 1989.

[142] D. R. Radev, E. Hovy, and K. McKeown. Introduction to the special issue on summarization. *Comput. Linguist.*, 28(4):399–408, 2002.

[143] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248, 2005.

[144] K. Rajaraman and A.-H. Tan. Topic detection, tracking, and trend analysis using self-organizing neural networks. In *PAKDD*, pages 102–107, 2001.

[145] R. Ramakrishnan and A. Tomkins. Toward a peopleweb. *Computer*, 40(8):63–72, 2007.

[146] M. Richardson and P. Domingos. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *NIPS*, pages 1441–1448, 2002.

[147] S. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of SIGIR'94*, pages 232–241, 1994.

[148] S. E. Robertson, S. Walker, S. Jones, M. M.Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. K. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*, pages 109–126, 1995.

[149] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494, 2004.

[150] S. Roy, D. Gevry, and W. M. Pottenger. Methodologies for trend detection in textual data mining. In *the Textmine '02 Workshop, Second SIAM International Conference on Data Mining*, 2002.

[151] C. Sagan. *Billions and Billions: Thoughts on Life and Death at the Brink of the Millennium*. New York: Random House, 1997.

[152] A. Shakery and C. Zhai. Smoothing document language models with probabilistic term count propagation. *Information Retrieval*, 11(2):139–164, 2008.

[153] C. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.

[154] C. Shannon. Prediction and entropy of printed english. *Bell Systems Technical Journal*, 30:50–64, 1951.

[155] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50, 2005.

[156] X. Shen, B. Tan, and C. Zhai. Ucair: a personalized search toolbar. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 681–681, 2005.

[157] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[158] L. Si and R. Jin. Adjusting mixture weights of gaussian mixture model via regularized probabilistic latent semantic analysis. In *PAKDD*, pages 622–631, 2005.

[159] X. Song, C.-Y. Lin, B. L. Tseng, and M.-T. Sun. Modeling and predicting personal information dissemination behavior. In *Proceeding of KDD '05*, pages 479–488, 2005.

[160] S. G. Sripada, E. Reiter, J. Hunter, and J. Yu. Generating english summaries of time series data using the gricean maxims. In *Proceedings of KDD '03*, pages 187–196, 2003.

[161] M. Steyvers and T. Griffiths. Probabilistic topic models. *Handbook of Latent Semantic Analysis*, 2007.

[162] M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. Griffiths. Probabilistic author-topic models for information discovery. In *Proceedings of KDD '04*, pages 306–315, 2004.

[163] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th international conference on World Wide Web*, pages 675–684, 2004.

[164] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 718–723, 2006.

[165] T. Tao and C. Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 162–169, 2006.

[166] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 449–456, New York, NY, USA, 2005. ACM Press.

[167] B. Tseng, J. Tatemura, and Y. Wu. Tomographic clustering to visualize blog communities as mountain views. In *WWW 2005 Workshop on the Weblogging Ecosystem*, 2005.

[168] T. van Dijk. Text and context. *Longman*, 1977.

[169] C. Wang, J. Wang, X. Xie, and W.-Y. Ma. Mining geographic knowledge using location aware topic model. In *GIR '07: Proceedings of the 4th ACM workshop on Geographical information retrieval*, pages 65–70, 2007.

[170] X. Wang and A. McCallum. A note on topical n-grams. In *University of Massachusetts Technical Report UM-CS-2005-071*, 2005.

[171] X. Wang and A. McCallum. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of KDD '06*, pages 424–433, 2006.

[172] X. Wang, C. Zhai, X. Hu, and R. Sproat. Mining correlated bursty topic patterns from coordinated text streams. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 784–793, 2007.

[173] X. Wei and W. B. Croft. Lda-based document models for ad-hoc retrieval. In *Proceedings of SIGIR '06*, pages 178–185, 2006.

[174] R. W. White and G. Marchionini. Examining the effectiveness of real-time query expansion. *Information Processing and Management (IPM)*, 43(3):685–704, 2007.

[175] J. Wiebe, T. Wilson, and C. Cardie. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation (formerly Computers and the Humanities)*, 39, 2005.

[176] S. K. M. Wong and Y. Y. Yao. A probability distribution model for information retrieval. *Inf. Process. Manage.*, 25(1):39–53, 1989.

[177] R. Wurman. Information anxiety. *Doubleday*, 1989.

[178] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *Proceedings of SIGIR' 99*, pages 254–261, 1999.

[179] J. Yi, T. Nasukawa, R. C. Bunescu, and W. Niblack. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *Proceedings of ICDM 2003*, pages 427–434, 2003.

[180] C. Zhai. Fast statistical parsing of noun phrases for document indexing. In *Proceedings of the fifth conference on Applied natural language processing*, pages 312–319, 1997.

[181] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of CIKM '01*, pages 403–410, 2001.

[182] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of ACM SIGIR'01*, pages 334–342, Sept 2001.

[183] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, 2001.

[184] C. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. In *Proceedings of KDD '04*, pages 743–748, 2004.

[185] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, 2004.

[186] D. Zhou, X. Ji, H. Zha, and C. L. Giles. Topic evolution and social interactions: how authors effect research. In *Proceedings of CIKM '06*, pages 248–257, 2006.

[187] D. Zhou, E. Manavoglu, J. Li, C. L. Giles, and H. Zha. Probabilistic models for discovering e-communities. In *Proceedings of WWW '06*, pages 173–182, 2006.

[188] D. Zhou and B. Schölkopf. Discrete regularization. *Semi-supervised learning*, pages 221–232, 2006.

[189] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.

[190] X. Zhu and J. D. Lafferty. Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning. In *ICML*, pages 1052–1059, 2005.

Author's Biography

Qiaozhu Mei was born in Sichuan, China. He graduated from Peking University in 2003 with a B.S. degree in computer science. Following the completion of his Ph.D. at the University of Illinois at Urbana-Champaign, Qiaozhu Mei works as an assistant professor at University of Michigan.