INTEGRATIVE TEXT MINING AND MANAGEMENT IN MULTIDIMENSIONAL TEXT DATABASES

BY

DUO ZHANG

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, 2012

Urbana, Illinois

Doctoral Committee:

Associate Professor ChengXiang Zhai, Chair & Director of Research
Professor Jiawei Han
Associate Professor Kevin C. Chang
Principal Scientist, Ashok N. Srivastava, NASA Ames Research Center

Abstract

As the text information grows explosively in today's multidimensional text databases, managing and mining this kind of databases is now playing an extremely important role in every domain. Different from traditional text mining tasks that target at single data sets, a text management system for a multidimensional database requires its text mining functions performed in different contexts specified by the structured dimensions, and the system should well support OLAP (online analytical processing) of the text information. This is a big challenge for most existing text mining techniques because of the efficiency and the scalability issues. On the other hand, the huge amount of text information in such databases also provides us an opportunity of acquiring new knowledge out of it, which could be super beneficial.

In this thesis, I identified three major types of functions that a text management system should support in order to analyze multidimensional text databases: (1) effective and efficient *digestion*: the system should support users to digest the text information in an OLAP environment based on domain knowledge; (2) flexible *exploration*: the system should allow users to flexibly explore the text information based on *ad hoc* information needs; (3) discovery *analysis*: the system should effectively analyze the text data with consideration of the associated non-textual data and mine knowledge underlying the text information. All of these functions are integrative analysis of the structured data and the unstructured text data within a multidimensional text database.

I proposed and studied different novel models and infrastructures to support all the above functions. First, I proposed a novel model called *Topic Cube* which combines the OLAP technology for traditional data warehouses with probabilistic topic modeling approaches for text mining. Given a topic hierarchy based on domain knowledge, a topic cube mines semantic topics accordingly and organizes the text information along with the topic hierarchy so that domain experts can quickly digest the text information in different granularity of topics and within different context.

Second, a novel infrastructure *MiTexCube* is proposed to flexibly support various kinds of online explorations, such as summarizing the content of text cells or comparing the content of documents across multiple text cells. The text content in a *MiTexCube* is stored as a compact representation called *micro-clusters* which make the online processing very efficient. Third, aiming at a special type of discovery analysis, comparative analysis on different text fields, I proposed a probabilistic topic mapping (PTM) model for mining two parallel text fields to discover latent topics and their associations. The model can be directly applied on multidimensional text databases with two parallel text fields. For multidimensional text databases with only one text field, the structured data can align two subsets of the data and form a parallel document collection so that meaningful knowledge can be mined by the proposed model.

Extensive experiments on multiple real world multidimensional text databases show that the proposed Topic Cube, *MiTexCube*, and PTM are all effective and efficient for digesting, exploring and analyzing multidimensional text databases. Since these techniques are all general, they can be applied to any multidimensional text databases in different application domains.

To my parents for all their love.

Acknowledgments

First and foremost, I wish to express my deepest gratitude to my advisor, Professor ChengXiang Zhai. Without his continuous guidance, support, and encouragement, this thesis would not have been possible. It's Cheng who brought me to the world of text mining research, taught me how to identify high impact research problems, and showed me how to conduct solid research. His keen insight, great passion, and rigorous attitude towards research deeply influenced me. Besides research, his personality and teaching philosophy also set up a great example of being an advisor for me. I enjoyed every discussion with him in the past five years. It's my great pleasure to work with him during my doctoral study.

I would like to thank all the other thesis committee members, Professor Jiawei Han, Professor Kevin C. Chang, and Dr. Ashok N. Srivastava, for their generous time, commitment, and their constructive suggestions on my thesis. Special thanks to Professor Jiawei Han for his long-term collaboration and support through my whole Ph.D. study in UIUC. Special thanks to Professor Kevin C. Chang for all his suggestions and for him attending my prelim exam in the midnight when he was visiting Singapore. Special thanks to Dr. Ashok N. Srivastava for his critical feedback and insightful comments on my thesis to make it solid.

I have been blessed to receive much help from many collaborators, colleagues, and friends. I would like to thank the members of the TIMan Group for all their valuable suggestions and help, especially, Tao Tao, Xuehua Shen, Jing Jiang, Xu Ling, Qiaozhu Mei, Xuanhui Wang, Bin Tan, Yue Lu, Alexander Kotov, V.G. Vinod Vydiswaran, Yuanhua Lv, Hyun Duk Kim, Hongning Wang, Mingjie Qian, Kurtis Wang, and Jing Zou. I would also like to thank many other members of the DAIS Group, especially, Hong Cheng, Deng Cai, Tao Cheng, Tianyi Wu, Jing Gao, Yizhou Sun, Peixiang Zhao, Xin Jin, Bo Zhao, Rui Li, Mianwei Zhou, Chi Wang, Ming Ji, and Quanquan Gu. I had so much fun and happy memories with their accompany.

Lastly but most importantly, I'm grateful to my parents for their endless and unreserved love, who have been encouraging and supporting me all the time. To them I dedicate this thesis.

This thesis was supported in part by NASA grants NNX08AC35A and NRA-NNH10ZDA001N; the U.S. National Science Foundation grants IIS-0842769, IIS-0713571, IIS-0905215, IIS-0713581, and CNS-1028381; and U.S. Air Force Office of Scientific Research MURI award FA9550-08-1-0265.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this information age, text data in multidimensional databases has grown explosively and has been a tremendously valuable data source for every domain, e.g. business domain, aviation safety research, and medical care. Since in general, it is desirable to analyze text data together with non-textual data, it is necessary to efficiently and effectively manage multidimensional text databases so that useful knowledge can be mined for all kinds of applications.

A **M**ulti**D**imensional **T**ext database (MDT database) is generally made up of *structured dimensions* and one or several *text dimensions*. Structured dimensions usually contain structured values, such as time and location, which can be viewed as the *context* for text of each record. A text dimension contains one or many documents which are regarded as the "content measure" of each record in an MDT database.

In Table 1.1, we show an example of the MDT database in ASRS [2], which is the world's largest repository of safety information provided by aviation's frontline personnel. The database has both structured data (e.g., time, airport, and light condition) and unstructured text data such as narratives about anomalous events written by pilots or flight attendants as illustrated in the table. A text narrative usually contains hundreds of words.

Another example from business domain is shown in Table 1.2. Besides the structure dimensions, there are two parallel text fields in the database. In the first text field, each record contains a set of reviews from customers in a month. In the second text field, it contains some internal business responses and solutions according to the customer feedback.

Mining such MDT databases is extremely useful for today's business intelligence and knowledge management. As a specific example, imagine how a product manager would like to analyze an MDT database and how text mining functions could assist her to achieve the goal. First, the product manager would like to *digest* how the customers comment or complain about different aspects of

Table 1.1: An example of multidimensional text database in ASRS

ACN	**Time**	**Airport**	$\cdots$	**Light**	**Narrative**
101285	199901	MSP	$\cdots$	Daylight	...The COMMENT ON RADIO DISCIPLINE...
101286	199901	CKB	$\cdots$	Night	...SHOULDN'T THE TWR CTLR TELL ME...
101291	199902	LAX	$\cdots$	Dawn	...WHEN ACFT SLOWED FOR APCH SPDS...

Table 1.2: An example of multidimensional text database from Camera Producers

Product	**Time**	**Location**	$\cdots$	**Review**	**Response**
Camera 50D	2009.1	Chicago	$\cdots$	Customer Reviews	Internal Business Responses
Camera 70D	2009.1	New York City	$\cdots$	Customer Reviews	Internal Business Responses
Camera 5D	2009.2	Seattle	$\cdots$	Customer Reviews	Internal Business Responses

a particular product during last month, e.g. the weight of Camera 50D, the color choices, and etc. An ideal text mining function should summarize all the customer reviews from last month into different aspects according to the product's specification, so that the manager can quickly digest all the information within minutes instead of spending hours on reading thousands of reviews to get the idea. Also, providing some representative reviews for each aspect summary would be very helpful. Second, the product manager may want to further *explore* the reviews, not necessarily according to the product's specification. For example, she may want to cluster the reviews into different groups and read representative reviews from each group. She may also have a specific query in her mind, e.g. a competitor camera, and want to know how customers compare these two products. All these explorations will help her better understanding customer's opinions. Next, the product manager would also like to do a comparative *analysis* on the two text fields, *i.e.* "Reviews" and "Responses", find out what are the general business solutions for different types of customer complaints, and then response to new customer feedback. Similar examples can also be found in aviation research domain. For example, with ASRS database, a researcher may want to digest the mentions of anomaly events according to specified categories, explore all the pilot reports based on some specific queries, and propose possible solutions to an anomaly event based on previous solutions if available.

All these text mining functions are very attractive and promising. However, many challenges remain to be solved for the current techniques in order to effectively and efficiently support all these functions. First, all these online analysis tasks of the MDT databases require high efficiency of the text mining techniques, and the function should be carried out online for any context specified

by the structured data. However, many data mining algorithms are inefficient if directly applied to process large amounts of data. For example, even the simplest clustering algorithm would take too much time to be practical for online analysis of a large number of text documents. Second, to support flexible exploration of the text data, an infrastructure should be designed general enough to support different types of text mining functions efficiently. However, there are not many this kind of infrastructures existing in literature, and most of them are designed for very specific purposes. Third, there is a lack of general techniques for performing comparative analysis of text data in the context of non-textual data. For example, it is unclear about how to simultaneously mine two parallel text fields with different granularities of topics based on demands and find out the underlining correlations among them, which is very crucial for comparative analysis on two text fields.

My thesis work addresses these three challenges, where I proposed and studied different new general algorithms and models to support the desired functions for analyzing MDT databases. An overview of the thesis is shown in Figure 1.1. The work can be divided into three synergistic major components, and each of them supports digestion, exploration, and analysis in an MDT database, respectively. As shown in the figure, the digestion function is supported by the *Topic Cube* model [59, 62], which combines traditional OLAP techniques [5, 12, 23] with topic modeling approach [26, 9] in text mining. The focus of the work is the efficiency and offline materialization strategies. *MiTexCube* [60, 61] is a novel infrastructure that aims at exploring MDT databases efficiently online. The focus of the infrastructure is efficiency and flexibility so that different kinds of text mining applications can be built based on it. Finally, a Probabilistic Topic Mapping (PTM) model [58] is proposed for comparative analysis on two parallel text fields. The focus of this study is the quality of the mined knowledge and the scalability of the proposed model.

Information Digestion with Topic Cube

An important technology to exploit today's data warehouses is the Online Analytical Processing (OLAP) technology [5, 12, 23], which enables flexible interactive analysis of multidimensional data in different granularities. Therefore, it's desirable to use OLAP alike technologies to digest information in an MDT database. However, traditional OLAP technologies, though capable of dealing

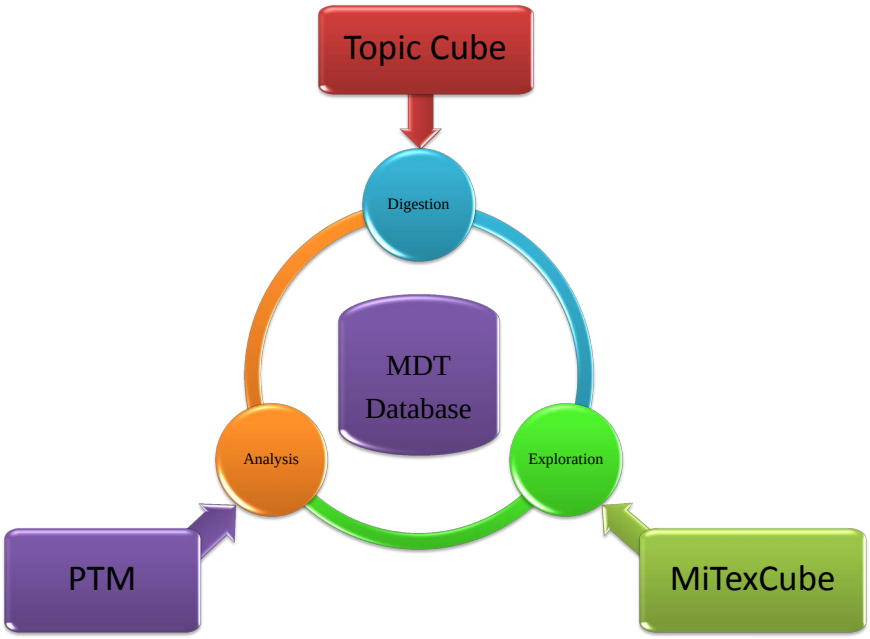


Figure 1.1: Thesis Overview: Three Major Components

with structured data, would face many challenges for analyzing unstructured text data. On the other hand, probabilistic topic modeling methods [26, 9] have been successfully applied in various text mining applications. Given prior knowledge about the categories of semantic topics embedded in the data, topic modeling approach is able to summarize the text documents according to the specified categories. Therefore, I proposed a novel model called *topic cube* which combines these two powerful techniques together to support the digestion function for MDT databases. I first formally defined a topic cube model, and described its star schema, the queries it supports, and the operations that can be done in a topic cube. Then, I solved a major challenge in materializing a topic cube, i.e. the offline computation efficiency. I studied several different strategies to materialize a topic cube and compared them with baseline methods. Experimental results on a real data set shows the advantages of the proposed strategies. Also, several applications that help digesting the text information are studied in the experiments, such as topic keywords comparison in different contexts and topic coverage comparison among different topics.

Information Exploration with *MiTexCube*

While a topic cube is capable of supporting content digestion in an MDT database, the flexibility of the model is limited because all the text documents are analyzed according to some specified topic categories. In many scenarios, however, users always have *ad hoc* information needs, which means pre-computing a set of semantic topics is not suitable anymore. For example, a user may want to explore the text documents by clustering them into different number of clusters. Or, she may type a short query and want to find out both relevant and representative documents for the query. Most current text mining techniques, however, are not efficient enough to support all these kinds of processing online, especially when the number of documents becomes very large. To solve both the flexibility and efficiency challenges, I defined a new infrastructure called Micro-Text-Cluster Cube (*MiTexCube* in short) which pre-computes similar documents into micro-clusters offline, and carries out online analysis based on these micro-cluster units. In the thesis, I first formally defined a *MiTexCube* infrastructure as well as its star schema. Then, I studied an efficient materialization algorithm for constructing a *MiTexCube* which flexibly balances the time-quality tradeoff for online applications and tries to save the disk cost as much as possible. An efficient updating algorithm for *MiTexCube* is also discussed. In the experiments, I examined three different online applications based on *MiTexCube* in terms of both efficiency and effectiveness, which showed the flexibility and efficiency advantage of using a *MiTexCube* infrastructure.

Comparative Analysis with PTM

Topic Cube and *MiTexCube* provide general support for users to digest and explore text data in a multidimensional database. However, to better support decision making, it is often necessary to go beyond supporting digestion and exploration to further support analysis of latent patterns in the database. To this end, in the third part of my thesis, I developed a novel Probabilistic Topic Mapping (PTM) model for discovering and comparing latent topical patterns embedded in text data. The PTM model can be applied to perform comparative analysis of two parallel text fields in an MDT database to extract topics from each field and discover their associations. MDT databases with multiple comparable text fields exist in many domains. For example, in business domain an

MDT may have "Problem" and "Solution" as two parallel text fields, while in medical care domain "Symptom" and "Treatment" always appear as parallel fields. Comparative analyses on these text fields would discover very valuable knowledge, *e.g.* possible business solutions to a certain type of problems or possible treatments for a certain kind of symptoms. Usually, there are different numbers of "Problem Topics" and "Solution Methods" in the two text fields. Thus, how to mine these two sets of topics (or methods) as well as their correlations are pretty challenging. In addition, a model which could scale well and work for millions of records is very much desirable. To address this need, I proposed this PTM model which can simultaneously mine two sets of topics from two parallel text fields and also capture the mappings between them. Experimental results show that PTM can effectively discover meaningful topics and the discovered topic mappings can be used to improve text matching when there is a vocabulary gap, demonstrating the effectiveness of the PTM model. We also implemented PTM with MapReduce on a large Hadoop cluster and tested its scalability on millions of records. The evaluation results show that the parallel implementation of PTM can scale up to process 3 millions of records within minutes on 200 mappers and 5 reducers.

The main contribution of this thesis is to systematically advance the state-of-the-art technology in analyzing multidimensional text databases, particularly in developing novel, general models and algorithms for supporting three types of information analysis functions, namely digestion, exploration, and analysis. All the algorithms are general and thus can be applied to any multidimensional text databases in any application domain.

The rest of the thesis is organized as follow: in Chapter 2, I will review related work in literature. In Chapter 3, 4, and 5, I will present our approaches for digestion (Topic Cube), exploration (*MiTexCube*), and analysis (PTM) in an MDT database. I will summarize my thesis and discuss possible future work in Chapter 6.

Chapter 2

Related Work

In this chapter, we will review related work in literature. The three major areas closely related to the thesis work are: (1) OnLine Analytical Processing (OLAP), (2) probabilistic topic modeling, and (3) document clustering. In particular, the first part of the thesis is a novel way to combine OLAP with probabilistic topic modeling; the second part is a novel infrastructure that leverages clustering to support efficient summarization of text data; and the third part is an extension of existing topic models for analyzing text in multidimensional text databases. Below we will briefly review the major work in each of these related areas.

2.1 Online Analytical Processing

Data warehouses are widely used in today's business market for organizing and analyzing large amounts of data. An important technology to exploit data warehouses is the OnLine Analytical Processing (OLAP) technology [5, 12, 23], which enables flexible interactive analysis of multidimensional data in different granularities. It has been widely applied to many different domains [22, 33, 50]. OLAP on data warehouses is mainly supported through data cubes [13, 14]. However, most of these work are not designed for supporting unstructured text data, which is the major difference from the first and the second part of this thesis.

2.2 Probabilistic Topic Modeling

Since the two fundamental work in Probabilistic Topic Modeling [26, 9] are published, a number of topic models have been extensively studied in recent years [7, 9, 26, 37] and have been successfully applied to a large range of text mining problems, such as hierarchical topic modeling [25, 8], opinion mining [51], information retrieval [53], social network analysis [34], spatiotemporal text mining [36],

sentiment analysis [35], and multi-stream bursty pattern finding [52]. All of these work showed that probabilistic topic models are very useful for analyzing latent topics in text data, and they are among the most effective text mining techniques. However, all the work in this line mostly focus on pure text data. The Topic Cube model studied in this thesis combines probabilistic topic modeling approach with OLAP technology to enable effective mining of both structured data and unstructured text data within a unified framework, which can be regarded as a novel application of such models to support OLAP on multidimensional text databases.

Besides the probabilistic topic models reviewed above, the most relevant work to the third part of this thesis focus on topic modeling on correlated text corpus, including authors and publications [48], citation documents [40], web pages and tags [43, 64], and poly-lingual corpus [39]. In [48], an author-topic model is proposed to model the words in a multi-author paper as the result of a mixture of each authors' topic mixture. The work [40] models documents and their citations by taking advantages of the link structures among documents and giving a better estimation of the embedded topics. The work [43, 64] study semantic topics on Web pages by correlating the content of web pages and their tags. Poly-lingual topic modeling in [39] aligns the documents that are loosely equivalent to each other but written in different languages.

To the best of our knowledge, no previous work has addressed the problem of mining parallel document collections to discover latent topic mappings. The main difference between the third part of this thesis, *i.e.* the proposed PTM model, and these existing topic models is that our model mines different sets of topics from two different but correlated document collections and at the same time analyze the mappings between these two sets of topics, while previous efforts either mine one set of topics from the publications with authors [48] or focus on mining one set of topics from one single corpus (with citation structure) [40]. [43], [64] and the bilingual study in [39] are similar to a special case of our model, i.e., the Alignment PTM, where we set the same number of topics for both source and target documents and let those topics have one-to-one correspondence. However, without imposing the one-to-one mapping constraint, PTM is more general. In addition, in all these works, a pair of correlated documents always have the same semantic content and share the same set of topics. But for parallel document collections in many domains like IT service and medical domain, we are more interested in revealing *different* sets of topics from the source set and

target set as well as how topics in one set are mapped to those in the other.

2.3 Document Clustering

Document clustering is also a well studied problem in text mining area. Numerous previous work have been done in this research area [24]. The materialization strategy used for materializing the *MiTexCube* in the second part of this thesis is inspired by the BIRCH algorithm described in [63], but the purpose of using micro-clusters is quite different. In *MiTexCube* we build micro-clusters in an OLAP environment, and the micro-clusters are used as coarse representations of the content of each local text cell. We do not maintain a global Clustering Feature tree as BIRCH does.

2.4 Other Related Work

Besides the three major related research areas discussed above, several other work are also relevant to the three components of this thesis.

2.4.1 OLAP for Text Analysis

In literature, there are some previous studies which have attempted to analyze text data in a relational database and support OLAP for text analysis. These studies can be grouped into four categories, depending on how they treat the text data.

Text as fact: In this kind of approaches, the text data is regarded as a fact of the data records. When a user queries the database, the fact of the text data, e.g. term frequency, will be returned. BlogScope [6], which is an analysis and visualization tool for blogosphere, belongs to this category. One feature of BlogScope is to depict the trend of a keyword. This is done by counting relevant blog posts in each time slot according to the input keyword and then drawing a curve of counts along the time axis. However, such an approach cannot support OLAP operations such as drill-down and roll-up on the text dimension, which the proposed topic cube would support. It's also different from those text mining functions that *MiTexCube* supports.

Text as character field: A major representative work in this group is [54], where the text data is treated as a character field. Given a keyword query, the records which have the most

relevant text in the field will be returned as results. For example, the following query (Location="Columbus", keyword="LCD") will fetch those records with location equal to "Columbus" and text field containing "LCD". This essentially extends the query capability of a relational database to support search over a text field. However, this approach cannot support OLAP on the text dimension either.

Text as categorical data: In this category, two similar work to ours are BIW [16] and Polyanalyst [4]. Both of them use classification methods to classify documents into categories and attach documents with class labels. Such category labels would allow a user to drill-down or roll-up along the category dimension, thus achieving OLAP on text. However, in [16], only high-level function descriptions are given with no algorithms given to efficiently support such OLAP operations on text dimension. Moreover in both work, the notion of cube is still the traditional data cube. Our topic cube differs from these two systems in that we integrate text mining (specifically topic modeling) more tightly with OLAP by extending the traditional data cube to cover topic dimension and support text content measures, which leads to a new cube model (i.e., topic cube), and it does not need any training data.

Text as component of OLAP There are also some related work using OLAP technology to explore unstructured text data in a multidimensional text database. In [46], the authors proposed a combination of keyword search and OLAP technique in order to efficiently explore the content of a multidimensional text database. The basic idea is to use OLAP technology to explore the search results from a keyword query, where some dynamic dimensions are constructed by extracting frequent and relevant phrases from the text data. In [32], a model called *text cube* is proposed, in which IR measures of terms are used to summarize the text data in a cell. Both of these work are different from Topic Cube in the way of digesting text information, and they are also different from *MiTexCube* which is a general infrastructure designed to support different online analysis efficiently.

2.4.2 Question and Answering

Another related research area to the third part of this thesis is Question-Answering (QA), and most related work can be found in TREC [19]. QA tasks are spread across different applications such as online forums [17], FAQ retrieval [47, 18], and email summarization [45]. In [55], the authors studied

using word-to-word translation probabilities to improve the retrieval models for QA archives. This is similar to one application of PTM for matching the source and target document. However, the difference is that our PTM learns "topic-to-topic" translation probabilities instead of word-to-word translations. In sum, while most of these studies aim at finding or constructing related answers to questions, our work focuses on analyzing semantic topics embedded in source and target corpus, as well as the mappings between these topics, and the mining results of our models can be used to enhance QA tasks.

Chapter 3

Topic Modeling for OLAP on Multidimensional Text Databases

Data warehouses are widely used in today's business market for organizing and analyzing large amounts of data. The Online Analytical Processing (OLAP) technology enables flexible interactive analysis of multidimensional data in different granularities, which is mainly supported through data cubes. As unstructured text data grows quickly in multidimensional text databases (MDT databases), it is more and more important to go beyond the traditional OLAP on structured data to also tap into the huge amounts of text data available to us for data analysis and knowledge discovery. In this Chapter, I propose a novel model called *Topic Cube*, which combines traditional OLAP technology for data warehousing with probabilistic topic modeling approach for text mining in order to support analysis in MDT databases. A Topic Cube adds a topic dimension to the tradition data cube which allows users to digest text information within a MDT database from different granularities of topics.

3.1 Introduction

As argued convincingly in [16], simultaneous analysis of both structured data and unstructured text data is needed in order to fully take advantage of all the knowledge in all the data, and will mutually enhance each other in terms of knowledge discovery, thus bringing more values to business. Unfortunately, traditional data cubes, though capable of dealing with structured data, would face challenges for analyzing unstructured text data.

As a specific example, consider ASRS [2], which is the world's largest repository of safety information provided by aviation's frontline personnel. The database has both structured data (e.g., time, airport, and light condition) and text data such as narratives about an anomalous event written by a pilot or flight attendant as illustrated in Table 1.1. A text narrative usually

contains hundreds of words.

This repository contains valuable information about aviation safety and is a main resource for analyzing causes of recurring anomalous aviation events. Since causal factors do not explicitly exist in the structured data part of the repository, but are buried in many narrative text fields, it is crucial to support an analyst to mine such data flexibly with both OLAP and text content analysis in an integrative manner. Unfortunately, the current data cube and OLAP techniques can only provide limited support for such integrative analysis. In particular, although they can easily support drill-down and roll-up on structured attributes dimensions such as "time" and "location", they cannot support an analyst to drill-down and roll-up on the text dimension according to some meaningful topic hierarchy defined for the analysis task (i.e. anomalous event analysis), such as the one illustrated in Figure 3.1.

In the tree shown in Figure 3.1, the root represents the aggregation of all the topics (each representing an anomalous event). The second level contains some general anomaly types defined in [1], such as "Anomaly Altitude Deviation" and "Anomaly Maintenance Problem." A child topic node represents a specialized event type of the event type of its parent node. For example, "Undershoot" and "Overshoot" are two special anomaly events of "Anomaly Altitude Deviation."

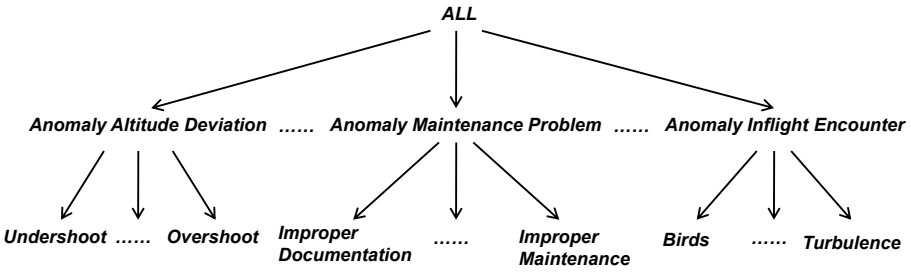


Figure 3.1: Hierarchical Topic Tree for Anomaly Event

Being able to drill-down and roll-up along such a hierarchy would be extremely useful for causal factor analysis of anomalous events. Unfortunately, with today's OLAP techniques, the analyst cannot easily do this. Imagine that an analyst is interested in analyzing *altitude deviation* problems of flights in Los Angeles in 1999. With the traditional data cube, the analyst can only pose a query such as (Time="1999",Location="LA") and obtain a large set of text narratives, which would have to be further analyzed with *separate* text mining tools.

Even if the data warehouse can support keyword queries on the text field, it still would not help the analyst that much. Specifically, the analyst can now pose a more constrained query (Time="1999",Location="LA", **Keyword="altitude deviation"**), which would give the analyst a smaller set of more relevant text narratives (i.e., those matching the keywords "altitude deviation") to work with. However, the analyst would still need to use a separate text mining tool to further analyze the text information to understand causes of this anomaly. Moreover, exact keyword matching would also likely miss many relevant text narratives that are about deviation problems but do not contain or match the keywords "altitude" and "deviation" exactly; for example, a more specific word such as "overshooting" may have been used in a narrative about an altitude deviation problem.

A more powerful OLAP system should ideally integrate text mining more tightly with the traditional cube and OLAP, and allow the analyst to drill-down and roll-up on the text dimension along a specified topic hierarchy in exactly the same way as he/she could on the location dimension. For example, it would be very helpful if the analyst can use a similar query (Time="1999",Location="LA", **Topic="altitude deviation"**) to obtain *all* the relevant narratives to this topic (including those that do not necessarily match exactly the words "altitude" and "deviation"), and then drill down into the lower-level categories such as "overshooting" and "undershooting" according to the hierarchy (and roll-up again) to change the view of the content of the text narrative data. Note that although the query has a similar form to that with the keyword query mentioned above, its intended semantics is different: "altitude deviation" is a *topic* taken from the hierarchy specified by the analyst, which is meant to match all the narratives covering this topic including those that may not match the keywords "altitude" and "deviation."

Furthermore, the analyst would also need to digest the content of all the narratives in the cube cell corresponding to each topic category and compare the content across different cells that correspond to interesting context variations. For example, at the level of "altitude deviation", it would be desirable to provide a summary of the content of all the narratives matching this topic, and when we drill-down to "overshooting", it would also be desirable to allow the analyst to easily obtain a summary of the narratives corresponding to the specific subcategory of "overshooting deviation." With such summaries, the analyst would be able to compare the content of narratives

about the encountering problem across different locations in 1999. Such a summary can be regarded as a content measure of the text in a cell.

This example illustrates that in order to integrate text analysis seamlessly into OLAP, we need to support the following functions:

Topic dimension hierarchy: We need to map text documents semantically to an arbitrary topic hierarchy specified by an analyst so that the analyst can drill-down and roll-up on the text dimension (i.e., adding text dimension to OLAP). Note that we do not always have training data (i.e., documents with known topic labels) for learning. Thus we must be able to perform this mapping without training data.

Text content measure: We need to summarize the content of text documents in a data cell (i.e., computing content measure on text). Since different applications may prefer different forms of summaries, we need to be able to represent the content in a general way that can accommodate many different ways to further customize a summary according to the application needs.

Efficient materialization: We need to materialize cubes with text content measures efficiently.

Although there has been some previous work on text database analysis [11, 29] and integrating text analysis with OLAP [28, 41], to the best of our knowledge, no previous work has proposed a specific solution to extend OLAP to support all these functions mentioned above. The closest previous work is the IBM work [16], where the authors proposed some high-level ideas for leveraging existing text mining algorithms to integrate text mining with OLAP. However, in their work, the cubes are still traditional data cubes, thus the integration is loose and text mining is in nature *external* to OLAP. Moreover, the issue of materializing cubes to efficiently handle text dimension has not be addressed.

In this thesis, we propose a new cube data model called *topic cube* to support the two key components of OLAP on text dimension (i.e., topic dimension hierarchy and text content measure) with a unified probabilistic framework. Our idea is to leverage probabilistic topic modeling [26, 9], which is a principled method for text mining, and combine it with OLAP. Indeed, PLSA and similar topic models have recently been very successfully applied to a large range of text mining problems. They are among the most effective text mining techniques. We propose Topic Cube to combine them with OLAP to enable effective mining of both structured data and unstructured text within

a unified framework.

Specifically, we will extend the traditional cube to incorporate the probabilistic latent semantic analysis (PLSA) model [26] so that a data cube would carry parameters of a probabilistic model that can indicate the text content of the cell. Our assumption is that we can use a probability distribution over words to model a topic in text. For example, a distribution may assign high probabilities to words such as "encounter", "turbulence", "height", "air", and it would intuitively capture the theme "encountering turbulence" in the aviation report domain. We assume all the documents in a cell to be word samples drawn from a mixture of many such topic distributions, and can thus estimate these hidden topic models by fitting the mixture model to the documents. These topic distributions can thus serve as content measures of text documents. In order to respect the topic hierarchy specified by the analyst and enable drill-down and roll-up on the text dimension, we further structure such distribution-based content measures based on the topic hierarchy specified by an analyst by using the concept hierarchy to impose a prior (preference) on the word distributions characterizing each topic, so that each word distribution would correspond to precisely one topic in the hierarchy. This way, we will learn word distributions characterizing each topic in the hierarchy. Once we have a distributional representation of each topic, we can easily map any set of documents to the topic hierarchy.

Note that topic cube supports the first two functions in a quite general way. First, when mapping documents into a topic hierarchy, the model could work with just some keyword description of each topic but no training data. Our experiment results show that this is feasible. If we do have training data available, the model can also easily use it to enrich our prior; indeed, if we have many training data and impose an infinitely strong prior, we essentially perform supervised text categorization with a generative model. Second, a multinomial word distribution serves well as a content measure. Such a model (often called unigram language model) has been shown to outperform the traditional vector space models in information retrieval [42, 57], and can be further exploited to generate more informative summaries if needed. For example, in [38], such a unigram language model has been successfully used to generate a sentence-based impact summary of scientific literature. In general, we may further use these word distributions to generate informative phrases [37] or comparative summaries for comparing content across different contexts [35]. Thus topic cube has potentially

many interesting applications.

Computing and materializing such a topic cube in a brute force way is time-consuming. So to better support the third function, we propose a heuristic algorithm to leverage estimated models for "component cells" to speed up the estimation of the model for a combined cell. Estimation of parameters is done with an iterative EM algorithm. Its efficiency highly depends on where to start in the parameter space. Our idea for speeding it up can be described as follows: We would start with the smallest cells to be materialized, and use PLSA to mine all the topics in the hierarchical topic tree from these cells, level by level. We then work on larger cells by leveraging the estimated parameters for the small cells as a more efficient starting point. We call this step as *aggregation along the standard dimensions*. In addition, when we mine the topics in the hierarchial tree from cells, we can also leverage the estimated parameters for topics in the lower level to estimate the parameters for topics in the higher level. We call this step as *aggregation along the topic dimension*. Our experiment results show that the proposed strategy, including both these two kinds of aggregations, can indeed speed up the estimation algorithm significantly.

3.2 Topic Cube as an Extension of Data Cube

The idea of topic cube is to extend the standard data cube by adding a topic hierarchy and probabilistic content measures of text so that we can perform OLAP on the text dimension in the same way as we perform OLAP on structured data. In order to understand this idea, it is necessary to understand some basic concepts about data cube and OLAP. So before we introduce topic cube in detail, we give a brief introduction to these concepts.

3.2.1 Standard Data Cube and OLAP

A data cube is a multidimensional data model. It has three components as input: *a base table, dimensional hierarchies, and measures*. A base table is a relational table in a database. A dimensional hierarchy gives a tree structure of values in a column field of the base table so that we can define aggregation in a meaningful way. A measure is a fact of the data.

Roll-up and *drill-down* are two typical operations in OLAP. Roll-up would "climb up" on a dimensional hierarchy to merge cells, while drill-down does the opposite and split cells. Other

OLAP operations include slice, dice, pivot, etc.

Two kinds of OLAP queries are supported in a data cube: *point query* and *subcube query*. A point query seeks a cell by specifying the values of some dimensions, while a range query would return a *set* of cells satisfying the query.

3.2.2 Overview of Topic Cube

A topic cube is constructed based on a *multidimensional text database (MTD)*, which we define as a multi-dimensional database with text fields. An example of such a database is shown in Table 1.1. We may distinguish a text dimension (denoted by TD) from a standard (i.e., non-text) dimension (denoted by SD) in a multidimensional text database.

Another component used to construct a topic cube is a *hierarchical topic tree*. A hierarchical topic tree defines a set of hierarchically organized topics that users are mostly interested in, which are presumably also what we want to mine from the text. A sample hierarchical topic tree is shown in Fig. 3.1. In a hierarchical topic tree, each node represents a topic, and its child nodes represent the sub-topics under this super topic. Formally, the topics are placed in several levels $L_1, L_2, \ldots, L_m$. Each level contains k_i topics, i.e. $L_i = (T_1, \ldots, T_{k_i})$.

Given a multidimensional text database and a hierarchical topic tree, the main idea of a topic cube is to use the hierarchical topic tree as the hierarchy for the text dimension so that a user can drill-down and roll-up along this hierarchy to explore the content of the text documents in the database. In order to achieve this, we would need to (1) map all the text documents to topics specified in the tree and (2) compute a measure of the content of the text documents falling into each cell.

As will be explained in detail later, we can solve both problems simultaneously with a probabilistic topic model called probabilistic latent semantics analysis (PLSA) [26]. Specifically, given any set of text documents, we can fit the PLSA model to these documents to obtain a set of latent topics in text, each represented by a word distribution (also called a unigram language model). These word distributions can serve as the basis of the "content measure" of text.

Since a basic assumption we make is that the analyst would be most interested in viewing the text data from the perspective of the specified hierarchical topic tree, we would also like these word

distributions corresponding well to the topics defined in the tree. Note that an analyst will be potentially interested in multiple levels of granularity of topics, thus we also would like our content measure to have "multiple resolutions", corresponding to the multiple levels of topics in the tree. Formally, for each level L_i, if the tree has defined k_i topics, we would like the PLSA to compute precisely k_i word distributions, each characterizing one of these k_i topics. We will denote these word distributions as θ_j, for $j = 1, ..., k_i$, and $p(w|\theta_j)$ is the probability of word w according to distribution θ_j. Intuitively, the distribution θ_j reflects the content of the text documents when "viewed" from the perspective of the j-th topic at level L_i.

We achieve this goal of aligning a topic to a word distribution in PLSA by using keyword descriptions of the topics in the tree to define a prior on the word distribution parameters in PLSA so that all these parameters will be biased toward capturing the topics in the tree. We estimate PLSA for each level of topics separately so as to obtain a multi-resolution view of the content.

This established correspondence between a topic and a word distribution in PLSA has another immediate benefit, which is to help us map the text documents into topics in the tree because the word distribution for each topic can serve as a model for the documents that cover the topic. Actually, after estimating parameters in PLSA we also obtain another set of parameters that indicate to what extent each document covers each topic. It is denoted as $p(\theta_j|d)$, which means the probability that document d covers topic θ_j. Thus we can easily predict which topic is the dominant topic in the set of documents by aggregating the coverage of a topic over all the documents in the set. That is, with $p(\theta_j|d)$, we can also compute the topic distribution in a cell of documents C as $p(\theta_j|C) = \frac{1}{|C|}\sum_{d\in C} p(\theta_j|d)$ (we assume $p(d)$ are equal for all $d \in C$). While θ_j is the primary content measure which we will store in each cell, we will also store $p(\theta_j|d)$ as an auxiliary measure to support other ways of aggregating and summarizing text content.

Thus essentially, our idea is to define a topic cube as an extension of a standard data cube by adding (1) a hierarchical topic tree as a topic dimension for the text field and (2) a set of probabilistic distributions as the content measure of text documents in the hierarchical topic dimension. We now give a systematic definition of the topic cube.

3.2.3 Definition of Topic Cube

Definition 3.2.1 *A **topic cube** is constructed based on a text database D and a hierarchical topic tree H. It not only has dimensions directly defined in the standard dimensions SD in the database D, but it also has a topic dimension which is corresponding to the hierarchical topic tree. Drill-down and roll-up along this topic dimension will allow users to view the data from different granularities of topics. The primary measure stored in a cell of a topic cube consists of a word distribution characterizing the content of text documents constrained by values on both the topic dimension and the standard dimensions (contexts).*

The star schema of a topic cube for the ASRS example is given in Fig. 3.2. The dimension table for the topic dimension is built based on the hierarchical topic tree. Two kinds of measures are stored in a topic cube cell, namely word distribution of a topic $p(w_i|topic)$ and topic coverage by documents $p(topic|d_j)$.

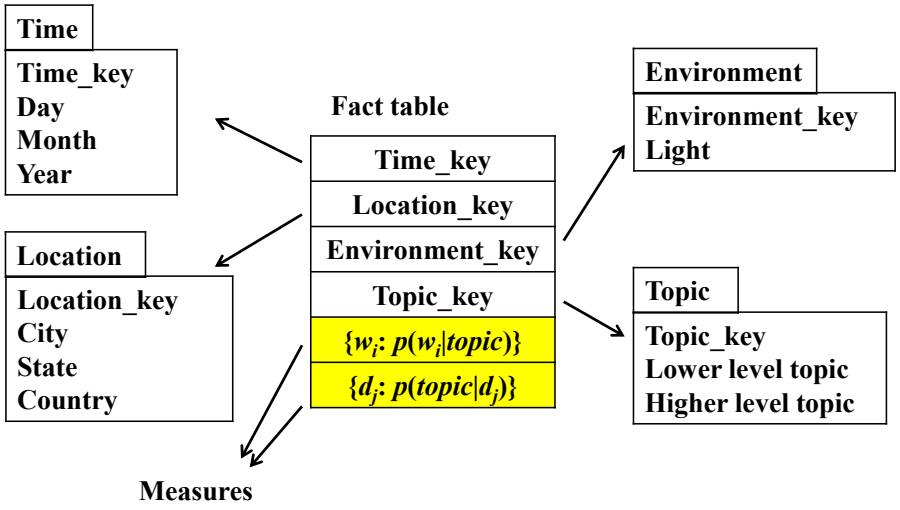


Figure 3.2: Star Schema of a Topic cube

Fig. 3.3 shows an example of a topic cube which is built based on ASRS data. The "Time" and "Location" dimensions are defined in the standard dimensions in the ASRS text database, and the topic dimension is added from the hierarchical tree shown in Fig. 3.1. For example, the left cuboid in Fig. 3.3 shows us word distributions of some finer topics like "overshoot" at "LAX" in "Jan. 99", while the right cuboid shows us word distributions of some coarse topics like "Deviation" at "LA" in "1999". In Fig. 3.4, it shows two example cells of a topic cube (with only word distribution

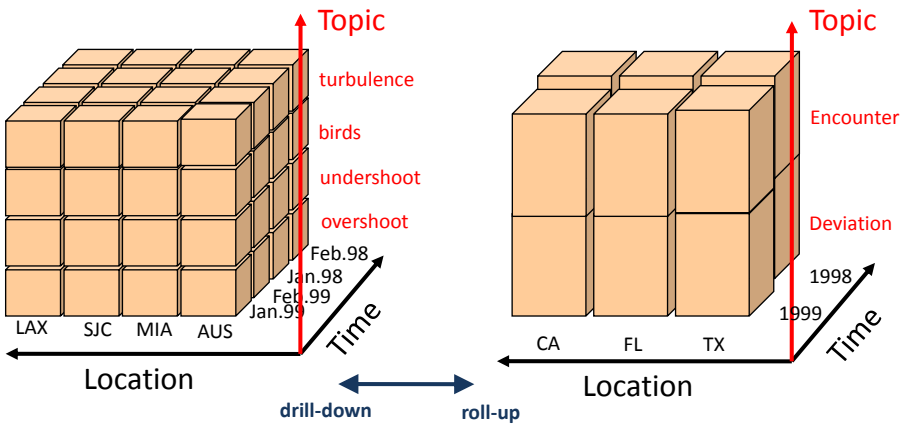


Figure 3.3: An example of a Topic Cube

measure) constructed from ASRS. The meaning of the first record is that the top words of aircraft equipment problem occurred in flights during January 1999 are (engine 0.104, pressure 0.029, oil 0.023, checklist 0.022, hydraulic 0.020, ...). So when an expert gets the result from the topic cube, she will soon know what are the main problems of equipments during January 1999, which shows the power of a topic cube.

Time	Anomaly Event	Word Distribution
1999.01	equipment	engine 0.104, pressure 0.029, oil 0.023, checklist 0.022, hydraulic 0.020, ...
1999.01	ground encounters	tug 0.059, park 0.031, pushback 0.031, ramp 0.029, brake 0.027, taxi 0.026, tow 0.023, ...

Figure 3.4: Example Cells in a Topic Cube

Query

A topic cube supports the following query: $(a_1, a_2, \ldots, a_m, t)$. Here, a_i represents the value of the i-th dimension and t represents the value of the topic dimension. Both a_i and t could be a specific value, a character "?", or a character "*". For example, in Fig. 3.3, a query ("LAX", "Jan. 99", t="turbulence") will return the word distribution of topic "turbulence" at "LAX" in "Jan. 99", while a query ("LAX", "Jan. 99", t="?") will return the word distribution of all the topics at "LAX" in "Jan. 99". If t is specified as a "*", e.g. ("LAX", "Jan. 99", t="*"), a topic cube will only return all the documents belong to (Location="LAX") and (Time="Jan. 99").

Operations

A topic cube not only allows users to carry out traditional OLAP operations on the standard dimensions, but also allows users to do the same kinds of operations on the topic dimension. The roll-up and drill-down operations along the topic dimension will allow users to view the data in different levels (granularities) of topics in the hierarchical topic tree. Roll-up corresponds to change the view from a lower level to a higher level in the tree, and drill-down is the opposite. For example, in Fig. 3.3, an operation:

Roll-up on Anomaly Event (from Level 2 to Level 1)

will change the view of topic cube from finer topics like "turbulence" and "overshoot" to coarser topics like "Encounter" and "Deviation". The operation:

Drill-down on Anomaly Event (from Level 1 to Level 2)

just does the opposite change.

3.3 Construction of Topic Cube

To construct a topic cube, we first construct a general data cube (we call it *GDC* from now on) based on the standard dimensions in the multidimensional text database D. In each cell of this cube, it stores a set of documents aggregated from the text dimension. Then, from the set of documents in each cell, we mine word distributions of topics defined in the hierarchical topic tree level by level. Next, we split each cell into $K = \sum_{i=1}^{m} k_i$ cells. Here, k_i is the number of topics in level i. Each of these new cells corresponds to one topic and stores its word distribution (primary measure) and the topic coverage probabilities (auxiliary measure). At last, a topic dimension is added to the data cube which allows users to view the data by selecting topics.

For example, to obtain a topic cube shown in Fig. 3.3, we first construct a data cube which has only two dimensions "Time" and "Location". Each cell in this data cube stores a set of documents. For example, in cell ("LAX", "Jan. 99"), it stores the documents belonging to all the records in the database which "Location" field is "LAX" and "Time" field is "Jan. 99". Then, for the second level of the hierarchical topic tree in Fig. 3.1, we mine topics, such as "turbulence", "bird", "overshoot",

and "undershoot", from the document set. For the first level of the hierarchical topic tree, we mine topics such as "Encounter" and "Deviation" from the document set. Next, we split the original cell, say ("LAX", "Jan. 99"), into K cells, e.g. ("LAX", "Jan. 99", "turbulence"), ("LAX", "Jan. 99", "Deviation") and etc. Here, K is the total number of topics defined in the hierarchical topic tree. At last, we add a topic dimension to the original data cube, and a topic cube is constructed.

Since a major component in our algorithm for constructing topic cube is the estimation of the PLSA model, we first give a brief introduction to this model before discussing the exact algorithm for constructing topic cube in detail.

3.3.1 Probabilistic Latent Semantic Analysis (PLSA)

Probabilistic topic models are *generative* models of text with latent variables representing topics (more precisely subtopics) buried in text. When using a topic model for text mining, we generally would fit a model to the text data to be analyzed and estimate all the parameters. These parameters would usually include a set of word distributions corresponding to latent topics, thus allowing us to discover and characterize hidden topics in text.

Most topic models proposed so far are based on two representative basic models: probabilistic latent semantic analysis (PLSA) [26] and latent Dirichlet allocation (LDA) [9]. While in principle both PLSA and LDA can be incorporated into OLAP with our ideas, we have chosen PLSA because its estimation can be done much more efficiently than for LDA. Below we give a brief introduction to PLSA.

Basic PLSA

The basic PLSA [26] is a finite mixture model with k multinomial component models. Each word in a document is assumed to be a sample from this mixture model. Formally, suppose we use θ_i to denote a multinomial distribution capturing the i-th topic, and $p(w|\theta_i)$ is the probability of word w given by θ_i. Let $\Theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$ be the set of all k topics. The log likelihood of a collection of text C is:

$$L(C|\Lambda) \propto \sum_{d \in C} \sum_{w \in V} c(w,d) \log \sum_{j=1}^{k} p(\theta_j|d) p(w|\theta_j) \tag{3.1}$$

where V is the vocabulary set of all words, $c(w, d)$ is the count of word w in document d, and Λ is the parameter set composed of $\{p(\theta_j|d), p(w|\theta_j)\}_{d,w,j}$.

Given a collection, we may estimate PLSA using the maximum likelihood (ML) estimator by choosing the parameters to maximize the likelihood function above. The ML estimator can be computed using the Expectation-Maximization (EM) algorithm [21]. The EM algorithm is a hill-climbing algorithm, and guaranteed to find a local maximum of the likelihood function. It finds this solution through iteratively improving an initial guess of parameter values using the following updating formulas (alternating between the E-step and M-step):

E-step:

$$p(z_{d,w} = j) = \frac{p^{(n)}(\theta_j|d)p^{(n)}(w|\theta_j)}{\sum_{j'=1}^{k} p^{(n)}(\theta_{j'}|d)p^{(n)}(w|\theta_{j'})} \tag{3.2}$$

M-step:

$$p^{(n+1)}(\theta_j|d) = \frac{\sum_w c(w, d)p(z_{d,w} = j)}{\sum_{j'} \sum_w c(w, d)p(z_{d,w} = j')} \tag{3.3}$$

$$p^{(n+1)}(w|\theta_j) = \frac{\sum_d c(w, d)p(z_{d,w} = j)}{\sum_{w'} \sum_d c(w', d)p(z_{d,w'} = j)} \tag{3.4}$$

In the E-step, we compute the probability of a hidden variable $z_{d,w}$, indicating which topic has been used to generate word w in d, which is calculated based on the current generation of parameter values. In the M-step, we would use the information obtained from the E-step to re-estimate (i.e., update) our parameters. It can be shown that the M-step always increases the likelihood function value, meaning that the next generation of parameter values would be better than the current one [21].

This updating process continues until the likelihood function converges to a local maximum point which gives us the ML estimate of the model parameters. Since the EM algorithm can only find a local maximum, its result is clearly affected by the choice of the initial values of parameters that it starts with. If the starting point of parameter values is already close to the maximum point, the algorithm would converge quickly. As we will discuss in detail later, we will leverage this property to speed up the process of materializing topic cube. Naturally, when a model has multiple local maxima (as in the case of PLSA), we generally need to run the algorithm multiple times, each time with a different starting point, and finally use the one with the highest likelihood.

PLSA Aligned to a Specified Topic Hierarchy

Directly applying PLSA model on a data set, we can extract k word distributions $\{p(w|\theta_i)\}_{i=1,\ldots,k}$, characterizing k topics. Intuitively these distributions capture word co-occurrences in the data, but the co-occurrence patterns captured do not necessarily correspond to the topics in our hierarchical topic tree. A key idea in our application of PLSA to construct topic cube is to *align* the discovered topics with the topics specified in the tree through defining a prior with the topic tree and using Bayesian estimation instead of the maximum likelihood estimator which solely listens to the data.

Specifically, we could define a conjugate Dirichlet prior and use the MAP (Maximum A Posteriori) estimator to estimate the parameters [49]. We would first define a prior word distribution $p'(w|\theta_j)$ based on the keyword description of the corresponding topic in the tree; for example, we may define it based on the relative frequency of each keyword in the description. We assume that it is quite easy for an analyst to give at least a few keywords to describe each topic. We then define a Dirichlet prior based on this distribution to essentially "force" θ_j to assign a reasonably high probability to words that have high probability according to our prior, i.e., the keywords used by the analyst to specify this topic would all tend to high probabilities according to θ_j, which further bias the distribution to attract other words co-occurring with them, achieving the purpose of extracting the content about this topic from text.

3.3.2 Materialization

As described in Section 3.3, to fully materialize a topic cube, we need to mine topics for each cell in the original data cube. As discussed earlier, we use the PLSA model as our topic modeling method. Suppose there are d standard dimensions in the text database D, each dimension has L_i levels ($i = 1, \ldots, d$), and each level has $n_i^{(l)}$ values ($i = 1, \ldots, d; l = 1, \ldots, L_i$). Then, we have totally $(\sum_{l=1}^{L_1} n_1^{(l)}) \times \cdots \times (\sum_{l=1}^{L_d} n_d^{(l)})$ cells need to mine if we want to fully materialize a topic cube. One baseline strategy of materialization is an exhaustive method which computes the topics cell by cell. However, this method is not efficient for the following reasons:

1. For each cell in *GDC*, the PLSA model uses EM algorithm to calculate the parameters of topic models. This is an iterative method, and it always needs hundreds of iterations before converge.

2. For each cell in *GDC*, the PLSA model has the local maximum problem. To avoid this problem and find the global maximization, it always starts from a number of different random points and selects the best one.

3. The number of cells in *GDC* could be huge.

On the other hand, based on the difficulty of aggregation, measures in a data cube can be classified into three categories: distributive, algebraic, and holistic [14]. As the measure in a topic cube is the word distributions of topics got from PLSA, we can easily see that it is a holistic measure. Therefore, there is no simple solution for us to aggregate measures from sub cells to super cells in a topic.

To overcome this problem, we propose to use a heuristic method, which is in a bottom-up manner, to materialize a topic cube more efficiently. On the other hand, it is also possible to materialize a topic cube by a top-down approach, if we prefer super or large cells rather than specific or small cells. In the following part of this section, we will first discuss two heuristic aggregation algorithms in the bottom-up strategy, which are the main approach we are using in our current study. Then, we will discuss a little bit about the top-down approach and compare it with the heuristic aggregation approach used in the bottom-up strategy.

Heuristic Materialization in Bottom-Up Strategy

The bottom-up strategy constructs a topic cube by first computing topics in small sub cells and then aggregate them to compute topics in large super cells. Our heuristic materialization algorithm contains two kinds of aggregations: *aggregation along the standard dimensions* and *aggregation along the topic dimension*. Either of these two aggregations can be used independently to materialize a topic cube. On the other hand, these two kinds of aggregations can also be applied together to further speed up the materialization. In this section, we first describe these two aggregations separately, and then we will discuss how to combine these two aggregations to materialize a topic cube in the next section.

The idea of aggregation along the standard dimensions is: when the heuristic method mines topics from documents of one cell in *GDC*, it computes the topics by first aggregating the word distributions of topics in its sub cells as a starting point and then using EM algorithm from this

starting point to get the local maximum result. In this way, the EM algorithm converges very quickly, and we do not need to restart the EM algorithm in several different random points. The outline of the heuristic aggregation along standard dimensions is shown in Table 3.1.

Table 3.1: Outline of Aggregation along Standard Dimensions

Suppose a topic cube has three standard dimensions A, B, C and a topic dimension T. The hierarchical topic tree H has n levels and each level L_i has k_i topics.

Step 1.

Build a general data cube *GDC* based on the standard dimensions, and each cell stores the corresponding document set.

Step 2.

· For each cell (a, b, c) in the base cuboid and a document set S_{abc} associated with it

· For each level L_i in H, where i is from 1 to $n-1$

· Estimate PLSA to mine k_i topics from the document set S_{abc} using the exhaustive method

Step 3.

· For each cell (a, b) in cuboid AB and a document set S_{ab} associated with it

· For each level L_i in H, where i is from 1 to $n-1$

· Estimate PLSA to mine k_i topics from S_{ab} by aggregating the same level of topics in all sub cells (a, b, c_j) of (a, b) in base cuboid

· Do similar aggregation in cuboid BC and CA

Step 4.

· Calculate topics for cells in cuboid A,B,C by aggregating from cuboid AB,BC,CA

Basically, in the first step of our algorithm, we construct a general data cube *GDC* based on the standard dimensions. Then in step 2, for each cell in the base cuboid, we use exhaustive method (starting EM algorithm from several random points and selecting the best local maximization) to mine topics from its associated document set level by level. In step 3 and step 4, we use our heuristic aggregation method to mine topics from cells in higher level cuboid. For example, when mining topics from cell (a, b) in *GDC*, we aggregate all the same level topics from its sub cells $(a, b, c_j)_{\forall c_j}$ in the base cuboid.

Specifically, suppose c_a is a cell in *GDC* and is aggregated from a set of sub cells $\{c_1, \ldots, c_m\}$, so that $S_{c_a} = \bigcup_{i=1}^{m} S_{c_i}$, where S_c is the document set associated with the cell S_c. For each level L_i in the hierarchical topic tree, we have got word distribution $\{p_{c_i}(w|\theta_1^{(L_i)}), \ldots, p_{c_i}(w|\theta_{k_i}^{(L_i)})\}$ for each sub cell c_i, and we are going to mine topics $\{\theta_1^{(L_i)}, \theta_2^{(L_i)}, \ldots, \theta_{k_i}^{(L_i)}\}$ from S_{c_a}. The idea of the heuristic method is when we apply EM algorithm for mining topics from S_{c_a}, we start from a *good starting point*, which is aggregated from word distributions of topics in its sub cells. The benefit

of a good starting point is that it can save the time for EM algorithm to converge and also save it from starting with several different random points.

The aggregation formulas are as follows:

$$p_{c_a}^{(0)}(w|\theta_j^{(L_i)}) = \frac{\sum_{c_i} \sum_{d \in S_{c_i}} c(w,d) p(z_{d,w} = j)}{\sum_{w'} \sum_{c_i} \sum_{d \in S_{c_i}} c(w',d) p(z_{d,w'} = j)} \tag{3.5}$$

$$p_{c_a}^{(0)}(\theta_j^{(L_i)}|d) = p_{c_i}(\theta_j^{(L_i)}|d), \qquad if\ d \in S_{c_i} \tag{3.6}$$

Intuitively, we simply pool together the expected counts of a word from each sub cell to get an overall count of the word for a topic distribution. An initial distribution estimated in this way can be expected to be closer to the optimal distribution than a randomly picked initial distribution.

Similarly, the idea of aggregation along the topic dimension is: for each cell in *GDC*, when we mine topics level by level in the hierarchical topic tree (from bottom to up), it computes the topics in the higher level of the hierarchical topic tree by first aggregating the word distributions of the topics in the lower level. The purpose is also trying to find a good starting point to run the EM algorithm so that it can converges quickly. The outline of the heuristic aggregation along the topic dimension can be described in Table 3.2.

Table 3.2: Outline of Aggregation along the Topic Dimension

Suppose a topic cube has several standard dimensions and a topic dimension T. The hierarchical topic tree H has n levels and each level L_i has k_i topics.

Step 1.

Build a general data cube *GDC* based on the standard dimensions, and each cell stores the corresponding document set.

Step 2.

· For each cell in *GDC* and a document set S associated with it

  · Estimate PLSA to mine k_{n-1} topics in the lowest level L_{n-1} from the document set S using exhaustive method

Step 3.

· For each cell in *GDC* and a document set S associated with it

  · For each level L_i in H, where i is from $n-2$ to 1

    · Estimate PLSA to mine k_i topics from the document set S by heuristic aggregating the topics mined in level L_{i+1}

After constructing a general data cube *GDC* based on the standard dimensions, we can use the heuristic aggregation along the topic dimension to mine a hierarchy of topics for each cell. For the topics in the lowest level, we just use PLSA to mine them from scratch. Then, we can mine topics in a higher level in the hierarchical topic tree by aggregating from the lower level topics.

Specifically, suppose one cell c has a set of documents S associated with it, and we have got word distributions for topics $\{\theta_1^{(L+1)}, \theta_2^{(L+1)}, \ldots, \theta_{k_{L+1}}^{(L+1)}\}$ in level $L+1$. Now we want to calculate the word distributions for topics $\{\theta_1^{(L)}, \theta_2^{(L)}, \ldots, \theta_{k_L}^{(L)}\}$ in level L. Here, each topic in level L has some subtopics (children nodes) in level $L+1$, i.e.:

$$Children(\theta_i^{(L)}) = \{\theta_{s_i}^{(L+1)}, \theta_{s_i+1}^{(L+1)}, \ldots, \theta_{e_i}^{(L+1)}\}, \forall 1 \leq i \leq k_L,$$
$$where \bigcup_{i=1}^{k_L} \{\theta_{s_i}^{(L+1)}, \ldots, \theta_{e_i}^{(L+1)}\} = \{\theta_1^{(L+1)}, \ldots, \theta_{k_{L+1}}^{(L+1)}\}$$

To get a good starting point to run the EM algorithm for estimating parameters, we can use the following formulas:

$$p_c^{(0)}(w|\theta_i^{(L)}) = \frac{\sum_{j' \in \{\theta_{s_i}^{(L+1)}, \ldots, \theta_{e_i}^{(L+1)}\}} \sum_{d \in S} c(w,d) p(z_{d,w}=j')}{\sum_{w'} \sum_{j' \in \{\theta_{s_i}^{(L+1)}, \ldots, \theta_{e_i}^{(L+1)}\}} \sum_{d \in S} c(w',d) p(z_{d,w'}=j')}$$

$$p_c^{(0)}(\theta_i^{(L)}|d) = \sum_{j' \in \{\theta_{s_i}^{(L+1)}, \ldots, \theta_{e_i}^{(L+1)}\}} p(\theta_{j'}|d), \qquad \forall d \in S \tag{3.7}$$

The intuition is similar: we pool together the expected counts of a word from each sub topic to get an overall count of the word for calculating their super topic's word distribution. After the initial distribution is calculated, we can run the EM algorithm until it converges, and it's expected to be quicker than a random initialization.

As described above, the two kinds of heuristic aggregations can be used independently to speed up the materialization of a topic cube. In fact, these two aggregations can be combined together to further improve the efficiency of materializing a topic cube. According to the order of the two aggregations in their combination, the combination strategy can have two forms:

(1) First aggregating along the topic dimension and then aggregating along the standard dimensions: For each cell in the base cuboid of a general data cube *GDC*, we exhaustively mined the topics in the lowest level of the hierarchical topic tree using PLSA. Then, we aggregate along the topic dimension and get all the topics for cells in the base cuboid. After that, we can use the heuristic aggregation along the standard dimensions to materialize all the other cells to construct a topic cube. This combination can be illustrated by Table 3.1 where we change Step 2 into "using the aggregation along the topic dimension to mine topics for each cell in the base cuboid".

(2) First aggregating along the standard dimensions and then aggregating along the topic dimension: For each cell in the base cuboid of a general data cube *GDC*, we exhaustively mined the topics in the lowest level of the hierarchical topic tree using PLSA. Then, we aggregate along the standard dimensions and get the lowest level topics for all the cells. After that, for each cell we can use the aggregation along the topic dimension to mine all the other level topics in the hierarchial topic tree. This combination can be illustrated by Table 3.2 where we change Step 2 into "using the aggregation along the standard dimensions to mine the lowest level topics for each cell in *GDC*".

The common part of these two different combinations is that they both need to use PLSA to exhaustively mine the topics in the lowest level of the hierarchical topic tree for all the cells in the base cuboid of *GDC*. After that, they will go along different directions. One interesting advantage of the second combination is that it can be used for materializing a topic cube in a parallel way. Specifically, after we get the lowest level topics for each cell in *GDC*, we can carry out the aggregation along the topic dimension for each cell in *GDC* independently. Therefore, Step 3 in Table 3.2 can be done in parallel.

Partial Materialization in Top-Down Strategy

Another possible strategy to materialize a topic cube is to compute it in a top-down manner. Specifically, this strategy first mine all the topics in the hierarchical tree from the largest cell (also called apex cell) in *GDC*. Then, when it computes the topics in the sub cells of the apex cell, it uses the computed word distributions of topics in the apex cell as starting points and mine topics in the sub cells individually. After the sub cells of the apex cell is materialized, it will compute the sub sub cells of the apex cell, and the word distributions of topics in their super cells will be used as starting points. This process will be continued iteratively until the whole topic cube is materialized. For example, if we want to materialize a topic cube as shown in Fig. 3.3 in a top-down strategy, we first mine all the topics in the apex cell of *GDC*, which is (Location="*", Time="*"). Then, we use the word distribution of the mined topics as starting points to mine topics in its sub cells, like (Location="CA", Time="*") and (Location="*", Time="1999"). After that, we can use the word distribution of topics in a new computed cell, like (Location="CA", Time="*"),

as the starting points to mine topics in its sub cells, like (Location="CA", Time="1999") and (Location="CA", Time="1998"). Note that if we do not use the word distribution of topics in a super cell as starting points when mining topics in sub cells, this strategy becomes an exhaustive top-down materialization method.

Compared with the bottom-up strategy, the advantage of the top-down strategy is: the materialization of a topic cube starts from the largest cell rather than the smallest cells in *GDC*. In this sense, we can stop materializing a topic cube at a certain level of cuboid in *GDC* if we believe that the current cuboid is too specific to be mined. This is reasonable because of two facts. First, when a cell is very specific, the number of documents contained in this cell will be small, which means this cell does not need to be mined or can be mined online. Second, in most cases users are interested in analyzing large and general cells in a cube rather than very specific cells. For example, suppose we have more than 20 standard dimensions in a *GDC*. When a user inputs a query $(a_1, a_2, \ldots, a_{20}, t)$, she may only specify a small number of a_i's value, and all the other dimensions will be set as "*". Indeed, it is always difficult for users to specify the values of all the dimensions in a data cube. Therefore, in a top-down strategy, not all the cells in *GDC* need to be materialized. This can also save a lot of disk cost of a topic cube.

On the other hand, the disadvantage of the top-down strategy is: to use the word distributions of topics in a super cell as the starting points when mining topics in its sub cells is not a good choice. Since a super cell is always made of a number of sub cells, the topics embedded in the documents of one single sub cell could be very different from the super cell. For example, if one sub cell A contains much smaller number of documents than its super cell, the topics mined in this super cell are mostly affected by its other sub cells, and the word distributions of these topics seems to be the same as a random point for sub cell A. But in the heuristic aggregations of the bottom-up strategy, the starting point (e.g. calculated by Eq. 3.6) during estimating the topics in a super cell is a weighted combination of topics in all the sub cells, which is much more meaningful.

3.3.3 Saving Storage Cost

One critical issue about topic cube is its storage. As discussed in Section 3.3.2, for a topic cube with d standard dimensions, we have totally $(\sum_{l=1}^{L_1} n_1^{(l)}) \times \cdots \times (\sum_{l=1}^{L_d} n_d^{(l)})$ cells in *GDC*. If there

are N topic nodes in the hierarchical topic tree and the vocabulary size is V, then we need at least store $(\sum_{l=1}^{L_1} n_1^{(l)}) \times \cdots \times (\sum_{l=1}^{L_d} n_d^{(l)}) \times N \times V$ values for the word distribution measure. This is a huge storage cost because both the number of cells and the size of the vocabulary are large in most cases.

There are three possible strategies to solve the storage problem. One is to reduce the storage by only storing the top k words for each topic. This method is reasonable, because in a word distribution of one topic, the top words always have the most information of a topic and can be used to represent the meaning of the topic. Although the cost of this method is the loss of information for topics, this strategy really saves the disk space a lot. For example, generally we always have ten thousands of words in our vocabulary. If we only store the top 10 words of each topic in the topic cube instead of storing all the words, then it will save thousands of times in disk space. The efficiency of this method is studied in our experiment part.

Another possible strategy is to use a general term to replace a set of words or phrases so that the size of the vocabulary will be compressed. For example, when we talk about an engine problem, words or phrases like "high temperature, noise, left engine, right engine" always appear. So it motivates us to use a general term "engine-problem" to replace all those correlated words or phrases. Such a replacement is meaningful especially when an expert only cares about general causes of an aviation anomaly instead of the details. But the disadvantage of this method is that it loses much detailed information, so there is a trade off between the storage and the precision of the information we need.

The third possible strategy is that instead of storing topics in all the levels in the hierarchical topic tree, we can select some levels of topics to store. For example, we can only store the topics in the odd levels or the topics in the even levels. The intuition is: suppose one topic's parent topic node has a word w at the top of its word distribution and at the same time its child topic node also has the same word w at the top of its word distribution, then it is highly probably that this topic also has word w at the top of its word distribution. In other words, for a specific topic, we can use the word distribution in both its parent topic and child topic to quickly induce its own word distribution. For another example, based on users' query history, we can also select those top popular topics to store, which means we only store the word distribution of mostly queried topics

for each cell. For those non-frequently queried topics, they may be just asked for a few times and within some specified cells, e.g. cells with a specified time period or a specified location. For these cases, we can just calculate the topics online and store the word distribution for these specified cells. By this strategy, it can help us to save a lot of disk spaces.

3.4 Experiments

In this section, we present our evaluation of the topic cube model. First, we compare the computation efficiency of our heuristic method with a baseline method which materializes a topic cube exhaustively cell by cell. Next, we are going to show several usages of topic cube to demonstrate its power.

3.4.1 Data Set

The data set we used in our experiment is downloaded from the ASRS database [2]. Three fields of the database are used as our standard dimensions, namely Time {1998, 1999}, Location {CA, TX, FL}, and Environment {Night, Daylight}. We use *A, B, C* to represent them respectively. Therefore, in the first step the constructed general data cube *GDC* has 12 cells in the base cuboid *ABC*, 16 cells in cuboids {*AB, BC, CA*}, and 7 cells in cuboids {*A, B, C*}. The summarization of the number of documents in each base cell is shown in Table 3.3.

Table 3.3: The Number of Documents in Each Base Cell

		CA	**TX**	**FL**
1998	Daylight	456	306	266
1998	Night	107	64	62
1999	Daylight	493	367	321
1999	Night	136	87	68

Three levels of hierarchical topic tree is used in our experiment, 6 topics in the first level and 16 topics in the second level, which is shown in Fig. 3.5. In real applications, the prior knowledge of each topic can be given by domain experts. For our experiments, we first collect a large number of aviation safety report data (also from ASRS database), and then manually check documents related to each anomaly event, and select top k ($k < 10$) most representative words of each topic as its prior.

Level 0	ALL					
Level 1	Equipment Problem	Altitude Deviation	Conflict	Ground Incursion	In-Flight Encounter	Maintain Problem
Level 2	Critical, Less Severe	Crossing Restriction Not Meet, Excursion From Assigned Altitude, Overshoot, Undershoot	Airborne, Ground, NMAC*	Landing Without Clearance, Runway	Turbulence, VFR* in IMC*, Weather	Improper Documentation, Improper Maintenance

⋆: NMAC-Near Midair Collision, VFR-Visual Flight Rules, IMC-Instrument Meteorological Conditions

Figure 3.5: Hierarchical Topic Tree used in the Experiments

3.4.2 Efficiency Comparison

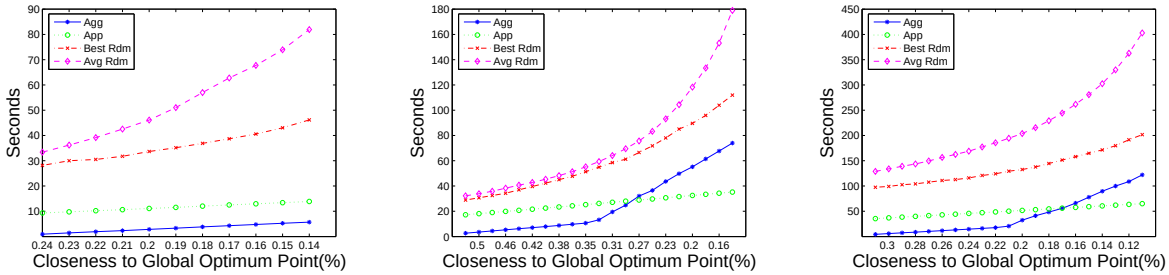
In this section, we evaluate the efficiency of the two kinds of heuristic aggregation we proposed in Section 3.3.2 separately. For each aggregation method, we compare three strategies of constructing a topic cube. (1) Heuristic aggregation method we proposed, either aggregation along the topic dimension or aggregation along the standard dimensions, and we use *Agg* to represent it. (2) An approximation method which only stores top k words in the word distribution of each topic, and we use *App* to represent it. The purpose of this method is to test the storage-saving strategy proposed in Section 3.3.3. For example, in the aggregation along standard dimensions when calculating topics from a document set in one cell, we use the same formula as in *Agg* to combine the word distributions of topics, with only top k words, in its sub cells and get a good starting point. Then, we initialize the EM algorithm with this starting point and continue it until convergence. Similarly, in the aggregation along the topic dimension, we also use only the top k words in the lower level topics to aggregate a starting point when we estimate the topics in the higher level. In our experiment, we set the constant k equal to 10. (3) The third strategy is the baseline of our method, which initializes the EM algorithm with random points, and we use *Rdm* to represent it. As stated before, the exhaustive method to materialize a topic cube runs EM algorithm by starting from several different random points and then select the best local maximum point. Obviously, if the exhaustive method runs EM algorithm M times, its time cost will be M times of the *Agg* method. The reason is every run of EM algorithm in *Rdm* has the same computation complexity as the *Agg* method. Therefore, it's no doubt that the heuristic aggregation method is faster than the exhaustive method. So, in our experiment, we use the *average performance* and *best run* of

the random method to compare the efficiency with the *Agg* method. The average performance is calculated by running the EM algorithm from M random points and then averaging the performance of these runs. The best run is the one which converges to the best local optimum point (highest log likelihood) among these M runs.

To measure the efficiency of these strategies, we look at how long it takes for these strategies to get to the same closeness to the global optimum point. Here, we assume that the convergence point of the best run of the M runs is the global optimum point. The experimental results are shown in Fig. 3.6. The upper three graphs show the efficiency comparison among the different strategies using aggregation along the standard dimensions, and the topics we computed in these three graphs are the 16 topics in the lowest level of the hierarchical topic tree in Figure 3.5. Each graph represents the result in one level of cuboid in the *GDC* cube, and we use one representative cell to show the comparison. The experiment on other cells have similar performance and can lead to the same conclusion. Similarly, the lower three graphs show the efficiency comparison using aggregation along the topic dimension, and the topics we computed are the 6 topics in the second level of the hierarchical topic tree.

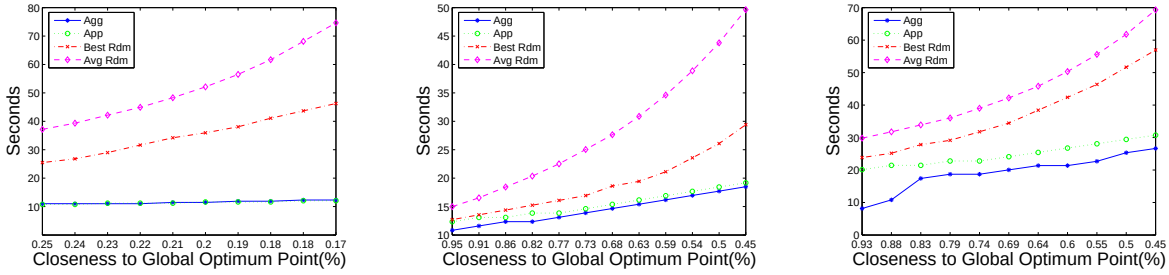
In the graph, *Best Rdm* represents the best run among those M random runs in the third strategy, and *Avg Rdm* represents the average performance of the M runs. The abscissa axis represents how close one point is to the global optimum point. For example, the value "0.24" on the axis means one point's log likelihood is 0.24% smaller than the log likelihood of the global optimum point. The vertical axis is the time measured in seconds. So a point in the plane means how much time a method needs to get to a certain closeness to the global optimum point. We can conclude that in all three cells, the proposed heuristic methods perform more efficiently than the baseline method, and this advantage of the heuristic aggregation is not affected by the scale of the document set. An interesting discovery is that the *App* method performs comparably with the *Agg* method, and in some cases it is even more stabler than *Agg*. For example, in Fig. 3.6 (b) and (c), although the *Agg* method starts from a better point than *App*, after reaching a certain point, the *Agg* method seems to be "trapped" and needs longer time than *App* to get further close to the optimum point.

Table 3.4 shows the log likelihood of the starting points of the three strategies. Here, the log



(a) Cell=(1999, CA, *) with 629 documents (b) Cell=(1999, *, *) with 1472 documents (c) Cell=(*, *, *) with 2733 documents

Using aggregation along the standard dimensions



(d) Cell=(1999, CA, *) with 629 documents (e) Cell=(1999, *, *) with 1472 documents (f) Cell=(*, *, *) with 2733 documents

Using aggregation along the topic dimension

Figure 3.6: Efficiency Comparison of Different Strategies

likelihood of the objective function is calculated by Eq. (3.1). This value indicates how likely the documents are generated by topic models, so it is the larger the better. In all the cells, both *Agg* and *App* strategies (in both two kinds of aggregations) have higher value than the average value of the *Rdm* strategy. This also assists our conclusion that the proposed heuristic methods are much easier to get to the optimum point than starting from a random point, thus need less time to converge.

3.4.3 Topic Comparison in Different Context

One major application of a topic cube is to allow users explore and analyze topics in different contexts. Here, we regard all the standard dimensions as contexts for topics. Fig. 3.7 shows four cells in the topic cube constructed on our experiment data. The column of "Environment" can be viewed as the context of the topic dimension "Anomaly Event". Comparing the same topic in different contexts will discover some interesting knowledge. For example, from the figure we can

Table 3.4: Comparison of Starting Points in Different Strategies

Aggregation along the Standard Dimensions			
Strategy	(1999, CA, *)	(1999, *, *)	(*, *, *)
Agg	-501098	-1079750	-2081270
App	-517922	-1102810	-2117920
Avg Rdm	-528778	-1125987	-2165459
Best Rdm	-528765	-1125970	-2165440
Aggregation along the Topic Dimension			
Strategy	(1999, CA, *)	(1999, *, *)	(*, *, *)
Agg	-521376	-1111400	-2135910
App	-524781	-1116220	-2144730
Avg Rdm	-528796	-1126046	-2165551
Best Rdm	-528785	-1126040	-2165510

see that the "landing without clearance" anomaly event has more emphasis on the words "light", "ils"(instrument landing system), and "beacon" in the context of "night" than in the context of "daylight". This tells experts of safety issues that these factors are most important for landing and are mentioned a lot by pilots. On the other hand, the anomaly event "altitude deviation: overshoot" is not affected too much by the environment light, because the word distribution in these two contexts are quite similar.

Environment	Anomaly Event	Word Distribution
daylight	landing without clearance	**tower** 0.075, **pattern** 0.061, **final** 0.060, **runway** 0.052, **land** 0.051, **downwind** 0.039
night	landing without clearance	**tower** 0.035, **runway** 0.027, **light** 0.026, **lit** 0.014, **ils** 0.014, **beacon** 0.013
daylight	altitude deviation: overshoot	**altitude** 0.116, **level** 0.029, **10000** 0.028, **f** 0.028, **o** 0.024, **altimeter** 0.023
night	altitude deviation: overshoot	**altitude** 0.073, **set** 0.029, **altimeter** 0.022, **level** 0.022, **11000** 0.018, **climb** 0.015

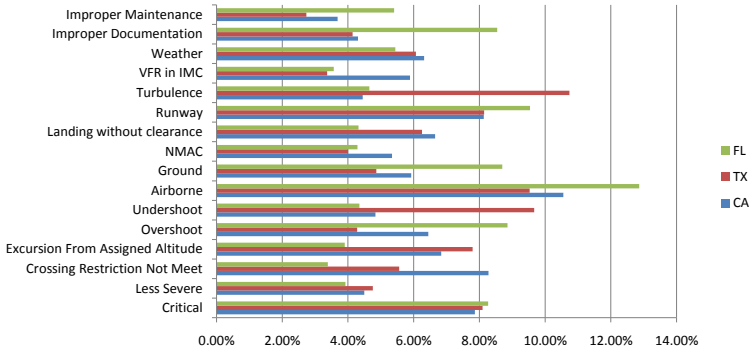
Figure 3.7: Application of Topic Cube in ASRS

3.4.4 Topic Coverage in Different Context

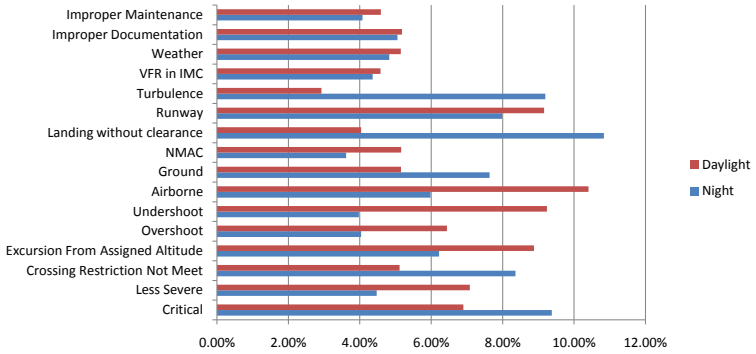
Topic coverage analysis is another function of a topic cube. As described above, one family of parameters in PLSA, $\{p(\theta|d)\}$, is stored as an auxiliary measure in a topic cube. The meaning of these parameters is the topic coverage over each document. With this family of parameters, we can analyze the topic coverage in different context. For example, given a context (Location="LA",

Time="1999"), we can calculate the coverage or proportion of one topic t by the average of $p(t|d_i)$ over all the document d_i in the corresponding cell in GDC. From another point of view, the coverage of one topic also reflects the severity of this anomaly event.

Fig. 3.8 shows the topic coverage analysis on our experiment data set. Fig. 3.8(a) is the topic coverage over different places and Fig. 3.8(b) is the topic coverage over different environment. With this kind of analysis, we can easily find out answers to the questions like: what is the most sever anomaly among all the flights in California state? What kind of anomaly is more likely to happen during night rather than daylight? For example, Fig. 3.8 helps us reveal some very interesting facts. Flights in Texas have more "turbulence" problems than in California and Florida, while Florida has the most sever "Encounter: Airborne" problem among these three places. And there is no evident difference of the coverage of anomalies like "Improper documentation" between night and daylight. This indicates that these kinds of anomalies are not correlated with environment factors very much. On the other hand, anomaly "Landing without clearance" obviously has a strong correlation with the environment.



(a) Place Comparison



(b) Environment Comparison

Figure 3.8: Topic Coverage Comparison among Different Contexts

3.4.5 Shaping Factor Analysis

Analyzing the shaping factors of human performance during flights plays an important role in aviation safety research. In pilot reports, the reporters tend to describe their physical factors, attitude, pressure, proficiency, preoccupation, and etc. So it's necessary to analyze all these factors and their correlations with anomalies during flight. With a topic cube, we can quantitatively evaluate the correlations between the anomaly events and the shaping factors of human performance in different context. Table 3.5 shows five different shaping factors as well as some examples and keywords of them.

Table 3.5: Examples and Keyword Lists of Shaping Factors

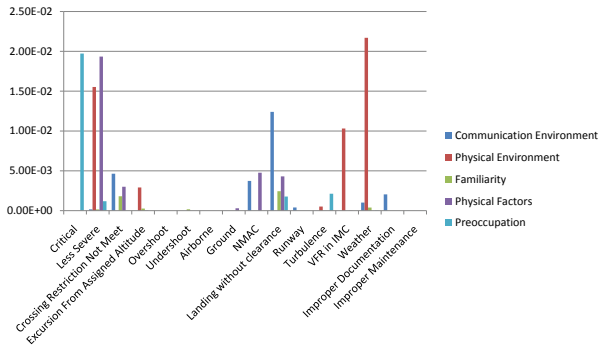
Shaping Factors	**Example**	**Keyword List**
Preoccupation	My attention was divided inappropriately.	distraction, attention, busy, emergency, realize, focus, declare
Communication Environment	We were unable to hear because traffic alert and collision avoidance system was very loud	communication, clearance, radio, frequency, hear, unreadable, wait
Familiarity	Both pilots were unfamiliar with the airport	unfamiliar, new, before, line, familiar, inexperienced, time
Physical Environment	This occurred because of the intense glare of the sun	weather, snow, cloud, wind, condition, ice, visibility
Physical Factors	I allowed fatigue and stress to cloud my judgment	fatigue, leg, hours, night, day, tire, rest

The keyword lists are extracted as follows: a human annotator is hired to annotate 1333 incident reports with 14 different shapers, where each report can be labeled with one or more shapers. Given the labeled reports, an algorithm [56] is used to compute the information gain of each unigram for each shaper. The top-k highest scored unigrams are selected as the keyword list for each shaper[1]. To quantitatively evaluate the correlation between a shaper S and an anomaly event A in a specific context C, we first find the word distribution in the cell specified by A and C in a topic cube. Then, the correlation value is calculated as the sum over all the keywords in S based on their probabilities in the word distribution.

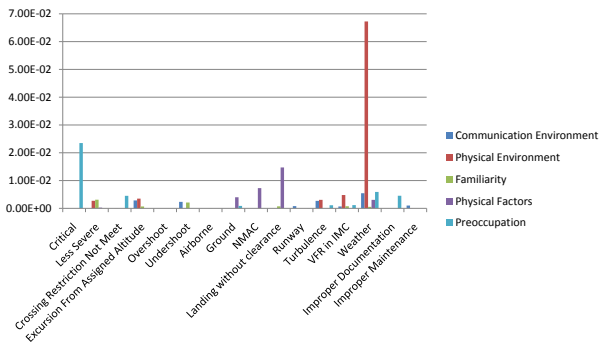
Figure 3.9 shows some examples of shaping factor analysis in different context with a topic cube. The x-axis in each graph represents different anomaly events, and the y-axis represents the

[1]We would like to thank Professor Vincent Ng from UT Dallas for providing us the keyword lists of shapers.

correlation between shaping factors and anomaly events (since the correlation is calculated as the sum over the probabilities of keywords of a shaper given the anomaly events, the value is relatively small as shown in the figure). From these graphs, we can find that "Physical Environment" is the main cause for anomaly events "Weather" and "VFR in IMC", no matter what the context is. This is reasonable from our common sense. On the other hand, if we compare the difference of the correlations among different contexts, we can also find some interesting things. For example, the shaping factor "Physical factors" causes more anomaly during night rather than daylight. The shaping factor "Communication Environment" causes the "Landing without clearance" anomaly event much more in Texas than in Florida, which suggests that airports or aircrafts in Texas may consider to improve their communication environment.

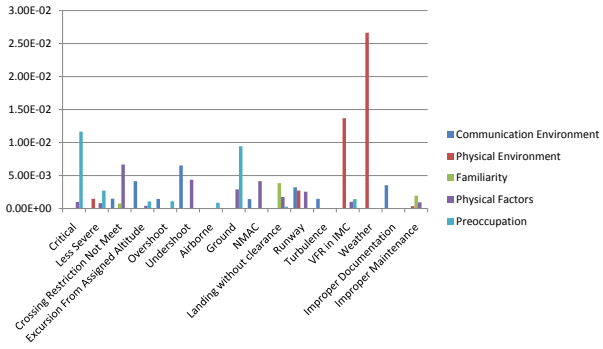


(a) Texas

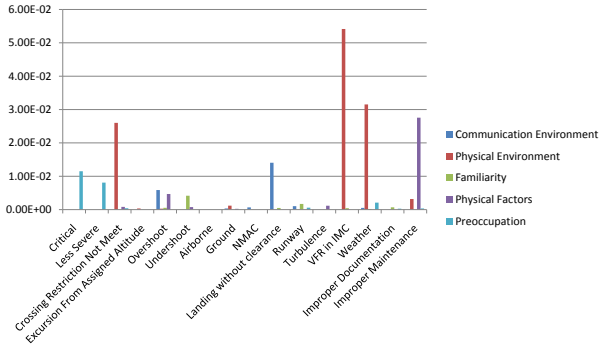


(b) Florida

Shaper Analysis in Different Places



(c) Night



(d) Daylight

Shaper Analysis in Different Environment

Figure 3.9: Shaper Analysis in Different Context

3.4.6 Accuracy of Categorization

In this experiment, we test how accurate the topic modeling method is for document categorization. Since we only have our prior for each topic without training examples in our data set, we do not compare our method with supervised classification. Instead, we use the following method as our baseline. First, we use the prior of each topic to create a language model ζ_j for each topic j. Then, we create a document language model ζ_d for each document after Dirichlet smoothing: $p(w|\zeta_d) = \frac{c(w,d)+\mu p(w|C)}{|d|+\mu}$, where $c(w,d)$ is the count of word w in document d and $p(w|C) = c(w,C)/|V|$ is the collection background model. Finally, we can use the negative KL-divergence [30] function to measure the similarity between a document d and a topic j: $S = -D(\zeta_j||\zeta_d) = \sum_w p(w|\zeta_j) \log \frac{p(w|\zeta_d)}{p(w|\zeta_j)}$. If one document d has a similarity score S higher than a threshold δ with a topic j, then it is classified into that topic. One the other hand, when we use the word distribution measure in a topic cube for categorization, we use the word distribution θ_j of topic j as its language model, and then compute the negative KL-divergence between θ_j and ζ_d to compute the similarity score of each topic j and document d.

Our experiment is conducted on the whole data set, and use the first level of topics in Fig. 3.5 as the target categories, i.e. we classify the documents into 6 categories. The gold answer we use is the "Anomaly Event" labels in ASRS data, which is tagged by pilots. Then we get the following recall-precision curves by changing the value of the threshold δ. As shown in Fig. 3.10, we can see that the curve of PLSA is above the baseline method. This means that PLSA would get better categorization result if we only have prior knowledge about topics.

3.5 Conclusions and Future Work

OLAP is a powerful technique for mining structured data, while probabilistic topic models are among the most effective techniques for mining topics in text and analyzing their patterns. In this study, we proposed a new data model (i.e., Topic Cube) to combine OLAP and topic models so that we can extend OLAP to the text dimension which allows an analyst to flexibly explore the content in text documents together with other standard dimensions in a multidimensional text database. Technically, we extended the standard data cube in two ways: (1) adopt a hierarchical topic tree

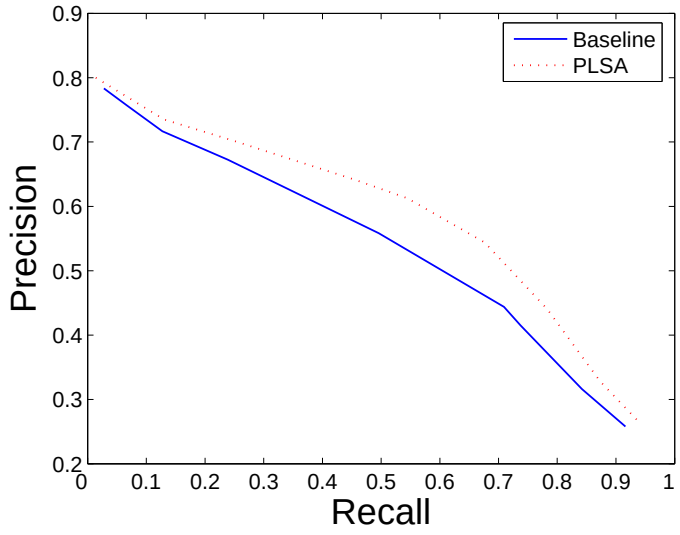


Figure 3.10: Comparison of Categorization Accuracy

to define a topic dimension for exploring text information, and (2) store word distributions as the primary content measure (and topic coverage probabilities as auxiliary measure) of text information. All these probabilities can be computed simultaneously through estimating a probabilistic latent semantic analysis model based on the text data in a cell. To efficiently materialize topic cube, we propose two kinds of heuristic aggregations which leverage previously estimated models in component cells or lower level topics to choose a good starting point for estimating the model for a merged large cell or higher level topics. Experimental results show that the heuristic aggregations are effective and topic cube can be used for many different applications.

Our work is only the first step in combining OLAP with text mining based on topic models, and there are many interesting directions for further study. First, it is necessary to further explore other possible strategies to materialize a topic cube efficiently, e.g. the top-down strategy discussed in Section 3.3.2. Second, it would be interesting to explore more application tasks to take advantage of the rich intermediate results of text analysis provide by a topic cube, and compare the topic cube with a baseline method, where OLAP on structured data and text mining are done separately, to verify its effectiveness in supporting an application task. Finally, our work only represents one way to combine OLAP and topic models. It should be very interesting to explore other ways to integrate these two kinds of effective, yet quite different mining techniques.

Chapter 4

MicroTextCluster Cube for Online Analysis of Text Cells

As large amount of unstructured text becomes available in multidimensional text databases, it is increasingly important to support efficient online analysis of text data. While a search engine is useful to satisfy a user's *ad hoc* information needs, allowing a user to retrieve relevant documents through a keyword query, it is inadequate for analysis of bulky text information, which is necessary in many online applications. For example, while it is easy for a user to find documents discussing opinions about iPhone in a review database based on a search engine, it is hard to compare opinions expressed in different time periods or by different user groups. In contrast, if we can manage text data together with structured data with attributes such as time and user groups in a multidimensional database, we would be able to flexibly explore text data corresponding to different combinations of time and user groups and compare opinions across different contexts. As discussed in last Chapter, a topic cube is capable of supporting content digestion in an MDT database. However, the flexibility of the model is limited because all the text documents are analyzed according to some specified topic categories. In many scenarios, when users have *ad hoc* information needs, pre-computing a set of semantic topics is not suitable anymore. In this Chapter, I propose and study a novel infrastructure called *MiTexCube*. In a *MiTexCube*, each text cell is materialized by *micro-clusters* which are grouped from similar documents offline. The purpose of building such a *MiTexCube* is to accelerate online analysis of text information by taking advantage of the micro-cluster units, which is way faster than calculations based on document units. I will use three different online applications to demonstrate the benefits of constructing a *MiTexCube*.

4.1 Introduction

Still consider the MDT database in ASRS as shown in Table 1.1. In many applications, we need to analyze the text information in such a multidimensional text database with consideration of structured data in the standard dimensions. To support such analysis in a general way, it has been proposed in recent work to construct a Text Cube [32] or a Topic Cube [59], which would enable an analyst to flexibly explore and analyze text cells, which are groups of text data corresponding to certain constraints on the standard dimensions.

Many interesting online analysis tasks can be done on top of text cells. For example, an expert may be interested in the major anomalous events within a specific context. So she forms a query like (Time="1999", Location="LA") and tries to digest the content of all the narratives associated with these specified time and location values. A desirable system would return a *summary* of the content in the specific text cell (e.g. clusters of documents with major content words in each cluster or a small set of representative documents) so that the expert does not need to read all the documents. In another scenario, the expert may be interested in a particular topic within a text cell, e.g. the anomalous events related to "altitude deviation" at "LA" during "1999". In this case, a set of documents, generated as a summary for a text cell *given a query topic*, should ideally be both relevant to the topic and representative in covering most content of the text data in the cell (many duplicates in selected documents will cause the summary cover only partial content of a text cell). Furthermore, the expert may also be interested in *comparing* the content of multiple cells, e.g. a group of cells with different locations, and it would be desirable for the system to generate a comparison of the content covered by all these returned cells to reveal some common topics discussed within these cells and the different coverage of these common topics in each cell. Similar application scenarios can also be found in many other domains such as product review analysis, IT service ticket investigation, and disease symptom diagnosis.

Since all these analysis tasks need to be done efficiently online, how to develop a *general* infrastructure to support all these tasks efficiently is a very interesting and challenging research question. Intuitively, we want to do as much offline pre-computing as possible to minimize the cost of online computation. However, there are two major technical challenges in implementing this general idea: (1) Many analysis tasks cannot be pre-specified in advance, making it impossible to pre-compute all

Table 4.1: An Example of a *MiTexCube*

Cell	Doc ID	Content	Micro-Text-Clusters
(Time=1999, Location=TX)	d_1	... due to stronger than forecasted winds and weather going ...	(weather 2.5, wind 1.2, ...), 3
	d_2	... I think that the weather, headwinds, shrinking dewpoint/temperature contributed to the fuel emergency ...	
	d_3	... After an hour, the weather had not much improved. We were in the clear for a bit and then hit another cloud bank ...	
	d_4	... so that if we saw the ARPT, we could land ...	(land 2.1, rule 0.9, ...), 2
	d_5	... we were in class G and the IFR rules tell us to land ...	

the answers or even partial answers. For example, query-specific summarization can only be done after seeing the query, thus a naive solution of computing and storing summaries of all the cells offline is simply not feasible. Indeed, it is a significant challenge to factor out the computation that can be done offline. (2) Different analysis tasks need different computations (e.g., summarization and topic comparison have different needs). It is unclear how to provide a *general* support for many such tasks to enable efficient online processing.

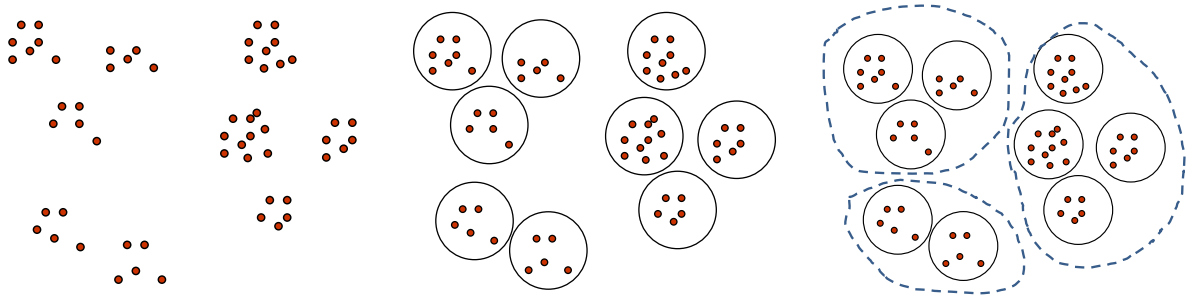
One possible solution to the two challenges is to build a global Clustering Feature (CF) tree as proposed in [63]. In this approach, all the documents in the multidimensional text database can be first clustered into a global CF tree offline. Then, online analysis can take advantage of the clusters stored in the CF tree to reduce the computational cost. However, such a global CF tree is not suitable for an OLAP scenario, because in an OLAP text analysis task we mainly focus on local contents of a text cube. When we change the context and do text analysis in different text cells, the rigid global clustering structure cannot serve well in various local cells, since the clustering results of documents in a local text cell could be very different from their clustering results in a global CF tree. For example, if we cluster all the reviews in a commercial text database, the global CF tree may cluster reviews based on different brands of products. But when we do OLAP analysis in a text cell of a certain location, the reviews within that text cell may be clustered according to different time periods. Similarly, if we do OLAP analysis in a text cell of a certain time period, the reviews may also be clustered according to different locations. So a global clustering structure based on brands is not suitable for analysis in different local text cells.

The recent work on Text Cube [32] proposed methods for analyzing a text cube by materializing

each text cell with vectors of documents. This approach can support several different analysis tasks, but it does not scale up well; indeed even a simple clustering analysis of the documents within one text cell is still expensive, especially when the number of documents is large.

In this study, we propose a new general infrastructure called *MicroTextCluster Cube* to organize text content in a multi-dimensional text database so as to support a variety of online text analysis tasks efficiently. To solve the two major challenges above, our key idea is to represent text contents of each **local cell** in a "compressed" way which can retain the essential semantic information in text, so that online operations can be supported efficiently by performing them on the compressed representation rather than the original representation.

Specifically, we cluster documents in each cell into *micro-clusters* which serve as a compact, though coarse, representation of the content in the cell. The set of documents in a micro-cluster can be regarded as a big "pseudo-document" with a compact representation. Since the number of micro-clusters in each cell is usually much smaller than the number of individual documents, it allows us to dynamically analyze any text cell (*e.g.*, clustering documents in a text cell) much more quickly based on the micro-clusters in the cell. Intuitively, the online computation effort is reduced substantially by offline micro-clustering of similar documents, as shown in Figure 4.1, where we see that online clustering can be done based on micro-clusters instead of the original documents. Since a common characteristic in many analysis tasks is that they focus more on the characteristics of groups of documents rather than the concrete content of each individual document, the micro-cluster model essentially captures and leverages this kind of redundant information to achieve a concise representation that enables many online analysis tasks to be done efficiently.



(a) Original Docs (b) Micro-clusters (c) Online Analysis

Figure 4.1: Illustration of micro-clusters and their uses for summarization.

We materialize a *MiTexCube* with a progressive strategy, which aims at both saving the disk cost of a *MiTexCube* and supporting efficient analysis of a large set of documents in a high level or large text cell with flexible tradeoff between efficiency and quality of analysis. Basically, one cell is materialized with micro-clusters only when the number of micro-clusters aggregated from its sub cells is too large to perform efficient online operation. In that case, the cell is materialized by re-clustering those micro-clusters in its sub cells into a small set of larger micro-clusters. During online analysis, we can either use the micro-clusters within the target cell or we can use more finer granularity micro-clusters aggregated from its sub cells if time cost is affordable. In an extreme case, we can also use single-document micro-clusters as the analysis units. Therefore, our approach makes it possible to control the efficiency-quality tradeoff through adjusting the resolutions of micro-clusters, and accommodate the different needs of different analysis tasks.

We also propose an algorithm to well maintain a *MiTexCube* when there are updates in the multidimensional text database, e.g. inserting new or deleting old documents. This is very important because these kinds of updates are very common in a data warehouse environment. Specifically, we use the Davies-Bouldin Index [20] to monitor the quality of micro-clusters within each text cell. When there are some updates in a text cell, we first simply update the corresponding micro-clusters based on the update in its sub cells. If the quality of the updated micro-clusters is below a threshold, then we need to recalculate the micro-clusters within the text cell as well as its super cells. Otherwise, we just keep the current updated micro-clusters. In this way, we can update a *MiTexCube* very efficiently while still maintaining high quality micro-clusters in it.

We represent each micro-cluster by a centroid vector of weighted terms, associated with certain statistics such as the size of the micro-cluster (*i.e.*, the number of documents inside the micro-cluster). Note that how to represent micro-clusters and how to form micro-clusters offline is quite flexible, as long as they can effectively compress the content of text cells in a reasonable way. We use weighted term vectors to represent micro-clusters and use k-means algorithm to form micro-clusters.

As a general infrastructure for representing text information in text cells in a compressed way, the micro-cluster text cube can potentially support many online analysis tasks efficiently. As case studies, we propose methods to leverage *MiTexCube* to support three common analysis tasks:

query-independent summarization, query-dependent summarization, and comparative analysis of text cells. We evaluate the proposed model and methods with the NASA:ASRS (Aviation Safety Report System) database and the DBLP [3] dataset. Experimental results show that (i) the proposed *MiTexCube* can be materialized efficiently with the proposed materialization algorithm with reasonable overhead in space, and (ii) the proposed cube structure can efficiently support summarization and comparative analysis of text cells, outperforming baseline methods that directly work on the documents in each cell without using micro-clusters, and it enables flexible tradeoff between efficiency and quality of analysis.

4.2 MicroTextCluster Cube

The main idea of *MiTexCube* is to speed up online analysis of text cells by doing as much preprocessing as possible during offline stage. Specifically, we preprocess the documents by generating a good number of micro-clusters to "compress" similar documents. These micro-clusters are materialized and stored in *selected* text cells. Since these micro-clusters can roughly represent the original documents, in the online stage, we can mostly work on the micro-clusters to carry out analysis of text cells quickly.

4.2.1 Definition of *MiTexCube*

Conceptually, *MiTexCube* extends a simple model, Document Cube. We thus first introduce the concept of document cube defined on a multidimensional text database.

Definition 4.2.1 ***Document Cube:*** *A document cube is a data cube built based on the standard dimensions of a multidimensional text database. The measure stored in each cell is a document set which is the union of the documents (records in the database) aggregated from its subcells.*

In general, a multidimensional text database is made of two parts: *standard fields* and a *text field*. The standard fields correspond to the attributes in a structured database (e.g., time, location) and can be viewed as the *context* of the associated text documents. Thus conceptually, the Document Cube allows us to naturally partition all the documents in the text field according to the combinations of values in the standard fields. Unfortunately, it is not feasible to store all the

document lists in cells. For example, in the apex cuboid, we need to store all the documents in its document list, which would be too expensive space-wise. Thus the measure in a document cube is only a "conceptual measure"; in practice, when a user inputs a query, a document cube would use the value(s) specified on the standard fields to fetch all the matching records in the database and return the union of the corresponding documents as the measure of the cell.

MiTexCube essentially extends Document Cube by storing an additional measure that captures all the micro-clusters in a cell.

Definition 4.2.2 ***MiTexCluster:*** *A micro text cluster (or MiTexCluster) is a coherent cluster of text documents that serves as a compressed representation of document content. These clusters are called micro-clusters because compared with the size of the corresponding cell that they represent, their sizes are relatively small, which ensures that the micro-clusters serve well as an approximation of the content in a cell for the purpose of analysis.*

Definition 4.2.3 ***MiTexCube:*** *A MiTexCube is a data model that extends a document cube to support efficient online analysis of text cells. Two kinds of measures are stored in cells of a MiTexCube. One is a document set aggregated from the base cells, which is the same as in a document cube. The other is either the statistics of a set of micro-clusters or a set of subcells from which the documents in the current cell can be efficiently computed based on aggregation.*

One important thing to be noted in Definition 4.2.3 is that information about micro-clusters (*i.e.*, "content measures") is stored in two different ways. We define the *MiTexCube* in this way in order to save the disk storage as much as possible. To better explain the idea behind this, we use an example to show the correspondence between cells and their measures in Table 4.2. From this table, we can see that there are mainly two types of cells. One is materialized with concrete micro-clusters, and we call this type of cells *concrete cells*. Examples of concrete cells are C_{71} and C_{100}. In their measures, we store five micro-clusters for each of them. Each micro-cluster contains its mean vector and the size of the cluster. In our study, each document is represented by a vector of weighted terms, and the weight for each term is the TF-IDF value of this term within the document [44]:

$$\vec{d} = (c_d(w_1) * idf_{w_1}, c_d(w_2) * idf_{w_2}, \ldots, c_d(w_V) * idf_{w_V})$$

Table 4.2: An example of the materialization of a *MiTexCube*

Cell ID	Type	Measure	Size
C_1	non-concrete	$\{d_1\}$	1
C_2	non-concrete	$\{d_2\}$	1
...	...	...	...
C_{70}	non-concrete	$\{C_1, C_2, \ldots, C_{10}\}$	10
C_{71}	concrete	$\{mean_1, 20\} \ldots \{mean_5, 30\}$	5
...	...	...	...
C_{99}	non-concrete	$\{C_{70}, C_{71}, \ldots, C_{76}\}$	56
C_{100}	concrete	$\{mean_1, 35\} \ldots \{mean_5, 32\}$	5
...	...	...	...

where $c_d(w_i)$ is the term frequency of word w_i in document d and idf_{w_i} is the inverse document frequency (IDF) of word w_i in the whole document set in the database. The mean vector of a micro-cluster is also a vector of weighted terms, and the weight for each term is the average weight for this term over all the documents that belong to this micro-cluster:

$$mean(mc_i) = \frac{1}{|mc_i|} \sum_{d \in mc_i} \vec{d}.$$

The other type of cells is materialized with a list of subcells, from which we can easily aggregate the micro-clusters in these subcells to form a set of micro-clusters for the current cell at the time of online processing. We call this type of cells *non-concrete cells*. For example, C_{99} is a non-concrete cell. Its measure contains a set of subcells, *i.e.*, $\{C_{70}, C_{71}, \ldots, C_{76}\}$. If we need to cluster the documents in cell C_{99}, we would fetch the micro-clusters contained in $\{C_{70}, C_{71}, \ldots, C_{76}\}$, and use them for clustering. In general, in order to save disk space, we would choose to not materialize a cell such as C_{99} as long as we can efficiently carry out analysis based on the micro-clusters contained in its subcells. However, had it been too expensive to do online analysis of the micro-clusters in $\{C_{70}, C_{71}, \ldots, C_{76}\}$, we would have further grouped these micro-clusters into larger micro-clusters and store them in the cell C_{99}, which would make it a concrete cell rather than a non-concrete cell.

In practice, storing the complete cell list in a non-concrete cell is still costly. Thus, we use a dimension and its level to indicate which set of subcells we should use for aggregation. For example, in Table 4.3, a cell (ID="laptop", Time="*", Location="*") can be either aggregated from subcells like {ID="laptop", Time="1st Quarter", Location="*"} or from subcells like {ID="laptop",

Table 4.3: An example of subcell selection

Cell ID	Subcell Set	Selection
(laptop, *, *)	(laptop, 1st Quarter, *)	{Time, Quarter Level}
	(laptop, 2nd Quarter, *)	
	(laptop, 3rd Quarter, *)	
	(laptop, 4th Quarter, *)	
	(laptop, Jan., *)	{Time, Month Level}
	(laptop, Feb., *)	
	...	
	(laptop, Dec., *)	
	(laptop, *, CA)	{Place, State Level}
	(laptop, *, TX)	
	...	
	(laptop, *, WA)	

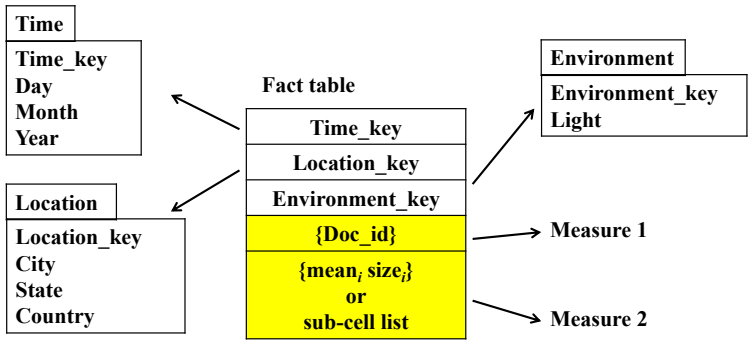


Figure 4.2: Star Schema of a *MiTexCube*

Time="*", Location="TX"}. So we use "{Time, Quarter Level}" or "{Place, State Level}" as a compact representation of the corresponding list of subcells. In the next section, we will discuss the criteria for choosing the dimension for aggregation.

The star schema of a *MiTexCube* is shown in Figure 4.2. In the schema, if we ignore Measure 2, it would become the star schema of a document cube. As we discussed above, Measure 1 in this schema is just a conceptual measure.

4.2.2 Progressive Materialization

Materialization of a *MiTexCube* means we need to precompute the micro-clusters offline and store the micro-clusters in the *MiTexCube*. A good materialization is important because (1) with sufficient materialization, the online analysis can be done efficiently; (2) the overhead of materialization should be reasonable.

Parameter Setting

There are two important parameters to be set for materializing a *MiTexCube*. One is the total number of micro-clusters K in each cell. A larger K will result in finer granularity of micro-clusters, so the result of online processing like clustering will be closer to that of clustering documents directly at the price of slower online processing. On the other hand, a smaller K will result in larger micro-clusters of documents in each cell, which can speed up online processing at the price of achieving coarser approximation of the content and not being able to summarize a cell at a finer granularity level of topics. Also, a small K will save the space cost of a *MiTexCube* and decrease the number of concrete cells. Thus there is an inherent tradeoff here among approximation accuracy, time efficiency of online summarization, and the space overhead for materializing a *MiTexCube* given different settings of K. So this parameter can be empirically set according to specific application needs. For example, in an application where users always want to summarize the content of a text cell into less than 10 groups, the parameter K could be set around 100 because 100 micro-clusters should be a good enough approximation (compared to document based clustering) for clustering the content into 10 groups, and storing 100 micro-clusters in each concrete cell is still affordable for the space.

The other important parameter is the total number of micro-clusters M that we can deal with efficiently for online processing. This parameter also controls the tradeoff between time efficiency and space overhead. If M is small, then the number of cells needed to be materialized will be large, thus more disk storage would be needed. On the other hand, if M is large, then the online processing will be more time consuming, but we would be able to save more space since the number of cells to be materialized would be smaller. Thus M provides a flexible way to control this tradeoff. Empirically, we optimize the value of M based on whether we can efficiently process M micro-clusters online. For example, if a machine can cluster at most 10000 micro-clusters into 10 groups within one minute during online processing, then we can set M as large as 10000.

An Example of the Overhead Estimation

The space overhead of the proposed materialization algorithm for *MiTexCube* is directly related to the number of concrete cells that we end up having since the overhead on the non-concrete cells is

Table 4.4: A theoretical study of materialization

Cuboid Level	Cells	docs per cell	micro-clusters per cell
0	1	V^N	$K * V$
1	$V * N$	V^{N-1}	K
...	...	...	...
$N-5$	$V^{N-5} * C_N^5$	V^5	K
$N-4$	$V^{N-4} * C_N^4$	V^4	$K * V$
$N-3$	$V^{N-3} * C_N^3$	V^3	K
$N-2$	$V^{N-2} * C_N^2$	V^2	V^2
$N-1$	$V^{N-1} * N$	V	V
N	V^N	1	1

insignificant. To better understand the space overhead, we perform some theoretical analysis using a simple example case.

Suppose we have a multidimensional text database with N standard dimensions, and without loss of generality, we assume N is an even number here. We further assume that each dimension has only one level, and each level has V values. In such a case, if we construct a document cube based on this database, we will have totally $(V+1)^N$ text cells. Suppose for each base cell we have exactly one document in it, and in total we have V^N documents in the database. Now, assume that the values of K and M satisfy the following conditions $K * V < M < K * V^2$ and $V^2 < M < V^3$. Then, the materialization result is shown in Table 4.4. We can see that only cells in cuboid levels $N-3$, $N-5$, ..., 1 need to be materialized. So the number of fully materialized cells is less than $\frac{(V+1)^N+(V-1)^N}{2}$, which means that roughly speaking, only 50% cells are materialized. Similarly, if $K * V^2 < M < K * V^3$, then the number of fully materialized cells will be less than 33%. In the experiment section, we will present an empirical comparison of the number of concrete cells in a *MiTexCube* with different parameter settings.

Materialization Algorithm

Once the two parameters K and M are set, our algorithm for materializing a *MiTexCube* works in a bottom-up manner to progressively process each cell. The process is illustrated in Figure 4.3.

Specifically, we would start materializing the cube from the base cells, each of which contains only one document. As we aggregate a set of base cells into the next level of cube (*i.e.*, cuboid ABCD, ABDE, *etc.*.), we would test the number of documents in each cell. If the number is larger than the threshold M, we would group this set of documents into K micro-clusters, and store these

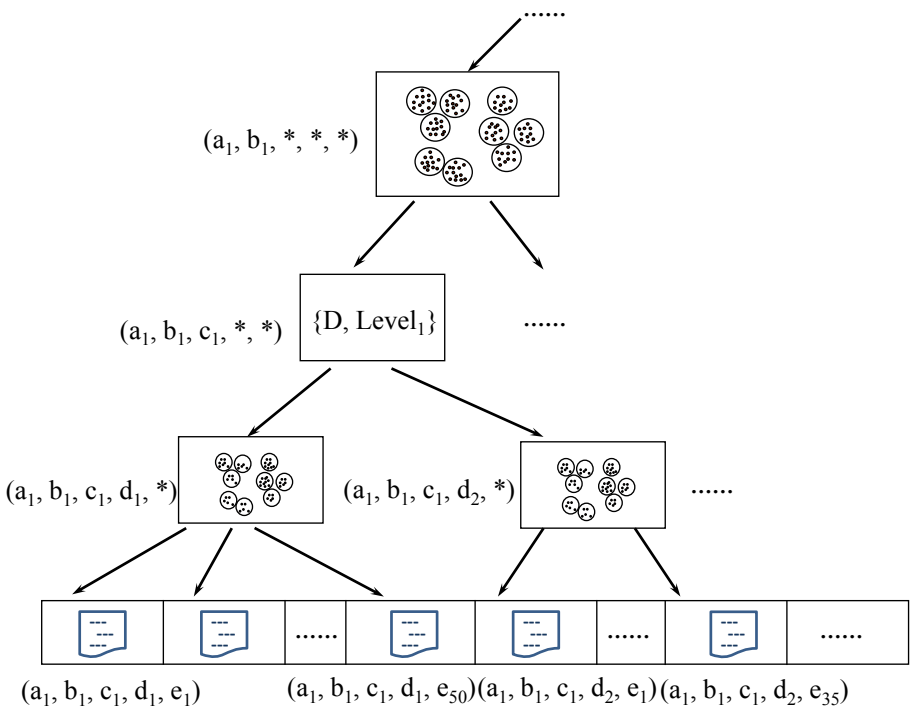


Figure 4.3: Materialization of a *MiTexCube*

micro-clusters as measures, as illustrated in cell $(a_1, b_1, c_1, d_1, *)$. Based on these micro-clusters, we can further aggregate into the next level of the cube (cuboid ABC, ABD, *etc.*). If the number of micro-clusters aggregated into one cell is no larger than the threshold M, we would only need to store a sub-cell list represented by the aggregation dimension and the level of this dimension, from which we will be able to aggregate the micro-clusters from the subcells, *e.g.*, cell $(a_1, b_1, c_1, *, *)$, thus saving the space needed to store the complete list of subcells. As shown in Table 4.3 and discussed in the previous section, there are several possibilities to aggregate subcells into a super cell. In our algorithm, we choose to store the dimension from which the subcells of the super cell would give the least number of micro-clusters; the rationale is to delay the need for re-clustering as much as we can, thus also saving more space. As we reach the next level of cube (*i.e.*, cuboid AB, BC, *etc.*), we first calculate the number of micro-clusters in each cell. For example, in cell $(a_1, b_1, *, *, *)$, the micro-clusters are aggregated from cell $(a_1, b_1, c_1, *, *)$, *etc.*Since cell $(a_1, b_1, c_1, *, *)$ is non-concrete, the micro-clusters aggregated from this cell are actually from its own subcells. Assuming that, at this time, the number of micro-clusters in cell $(a_1, b_1, *, *, *)$ is larger than the threshold M, we thus group these small micro-clusters into K larger micro-clusters, as shown in the figure.

In general, to group micro-clusters from subcells into larger micro-clusters, we may use any clustering algorithm. In our experiments, we use a k-means based algorithm to group those micro-clusters based on their means. One advantage of k-means is that we can stop at any iteration to obtain clustering results, and thus can flexibly trade off time with quality of clusters. For each small micro-cluster we have the mean and size of it. Therefore, we can use these statistics to calculate the statistics of the new micro-clusters. For example, suppose we group micro-clusters $mc_1, mc_2, ..., mc_l$ into a bigger micro-cluster. The mean of the new micro-cluster can be computed as

$$\frac{\sum_{i=1}^{l} mean(mc_i) * size(mc_i)}{\sum_{i=1}^{l} size(mc_i)}, \tag{4.1}$$

and the size of the new micro-cluster is

$$\sum_{i=1}^{l} size(mc_i) \tag{4.2}$$

The pseudo code of the materialization algorithm using k-means clustering is given in Algorithm 1. Here, variable min_{mc} is used to store the minimum number of micro-clusters aggregated into the current cell. Variables $min_{dimension}$ and min_{level} are used to store the corresponding dimension and level. Variable num_{mc} on line 8 represents the number of micro-clusters aggregated from subcells in the current dimension and level.

4.2.3 Update a *MiTexCube*

When there is any update in the multidimensional text database, we also need to update the *MiTexCube* infrastructure accordingly. Since we want to make the whole update process efficient, especially when the update of the database is small, we do not want to re-cluster a lot when updating the *MiTexCube*. Therefore, we propose an efficient updating algorithm which saves the efforts of re-clustering micro-clusters as long as the quality of the updated micro-clusters are still good.

Specifically, we associate each text cell with a quality measure which monitors how good its micro-clusters are. We adopt the Davies-Bouldin Index (DBI) proposed in [20] as the quality measure. Suppose we have totally K micro-clusters in a text cell, DBI is a cluster separation

Algorithm 1: Pseudo Code for *MiTexCube* Materialization

Input: A multidimensional text database with n standard fields and one text field
Output: Materialized *MiTexCube*

```
for cuboid in Base Cuboid to Apex Cuboid do
    foreach cell in current cuboid do
        min_mc = ∞;
        min_dimension = null;
        min_level = null;
        foreach aggregation dimension in current cell do
            foreach level in this dimension do
                Calculate num_mc in subcells aggregated from current dimension and level;
                if num_mc < min_mc then
                    min_mc = num_mc;
                    min_dimension = current dimension;
                    min_level = current level;
                end
            end
        end
        if min_mc > M then
            Regroup these micro-clusters into K larger micro-clusters, and store the mean
            and size of each new micro-cluster;
        else
            Store the min_dimension and min_level as measures;
        end
    end
end
```

measure which is defined as:

$$DBI = \frac{1}{K}\sum_{i=1}^{K}\max_{j:j\neq i}\frac{S_i + S_j}{M_{i,j}}$$

where S_i is a measure of scatter within cluster i, i.e. the average distance between each point within cluster i and the centroid of cluster i, and $M_{i,j}$ is a measure of separation between cluster i and cluster j, i.e. the distance between the two centroids. Therefore, a set of micro-clusters will have a good quality if they have a relatively low DBI. When we update the components of micro-clusters within a text cell, we calculate the DBI of the updated micro-clusters and compare it with the old DBI of that text cell. If the relative difference is not above a threshold δ, then we just keep these updated new micro-clusters. Otherwise, we need to re-cluster all the small micro-clusters within the text cell to get a group of high quality big micro-clusters.

The pseudo code of the update process is shown in Algorithm 2. The major part of why using

a DBI quality measure can save a lot of re-clustering efforts is shown from Line 2 to Line 2. In the algorithm description, when tc is a concrete cell, the K micro-clusters in tc (denoted as big-micro-clusters) are clustered from the micro-clusters in tc's sub cells (denoted as small-micro-clusters). Therefore, we can either remove a small-micro-cluster unit from a big-micro-cluster or insert one into the big-micro-cluster. The algorithm is pretty efficient when there is only a little update in the database. In an extreme case, if there are only a small number of new documents inserted, Algorithm 2 will just insert these documents into the corresponding micro-clusters, and there is no need to do any re-clustering during the update process.

4.3 Online Analysis of Text Cells

After a *MiTexCube* is materialized, we can carry out various kinds of online analysis based on this infrastructure. In this section, we discuss three representative online analysis tasks.

4.3.1 Standard (Neutral) Cell Summarization

Standard (*i.e.*, topic-neutral) cell summarization means to give analysts an overview of the content in any given text cell by grouping all the documents in that text cell into P different clusters, where P is the desired number of clusters specified by an analyst. Based on the *MiTexCube* model, one can efficiently generate such a standard cell summary by clustering the already formed micro-clusters instead of clustering all the documents from scratch. Specifically, assume one cell has in total MC micro-clusters ($MC > P$). We can use the mean vector of each micro-cluster as a data point (as if it were a document vector) and use the k-means algorithm to partition them into P clusters. When we cluster several micro-clusters into one big cluster, we can use Eq. (4.1) and Eq. (4.2) to update the mean and size of this big cluster. Thus algorithm-wise, our method for standard cell summarization is similar to re-clustering in the materialization algorithm except that we now generate fewer macro-clusters for the purpose of online analysis of a text cell's content.

Since the k-means method is an iterative method, the time complexity of the baseline method of clustering all the documents from scratch (which we denote by *GS-Base*) is $O(D * P * n)$, where D is the total number of documents to be clustered, n is the total number of iterations. With *MiTexCube*, the time complexity of our method (denoted by *GS-MC*) is $O(MC * P * m)$, where

Algorithm 2: Pseudo Code for *MiTexCube* Update

```
Input: A materialized MiTexCube and documents to be added and removed
Output: A new updated MiTexCube
for cuboid in Base Cuboid to Apex Cuboid do
    foreach cell tc in current cuboid do
        if there is no change in tc's sub cells then
            Continue;
        else if after update, tc has less than M documents then
            Leave tc as a non-concrete cell;
        else if tc is a non-concrete cell then
            Do Line 1 to Line 1 in Algorithm 1;
        else if tc is a concrete cell with K big-micro-clusters then
            Find all the old small-micro-clusters (in a low level cuboid, a small-micro-cluster
            could be a document) that made up these K big-micro-clusters, and find all the
            new small-micro-clusters generated from tc's sub cells after update;
            if any of these old small-micro-clusters are changed in tc's sub cells then
                Remove all these old small-micro-clusters and update those big-micro-clusters
                which contain them;
            end
            if the number of big-micro-clusters after deletion is less than K then
                Do Line 1 to Line 1 in Algorithm 1;
            else
                foreach new small-micro-clusters do
                    Find its nearest big-micro-cluster, put it in, and update the size and the
                    mean of this big-micro-cluster;
                end
                if new_DBI < (1 + δ) * old_DBI then
                    Continue;
                else
                    Regroup current small-micro-clusters into K big-micro-clusters;
                end
            end
        end
    end
end
```

MC is the number of micro-clusters and m is the number of iterations. When $MC \ll D$, we can expect that *GS-MC* should be much faster than *GS-Base* (the number of iterations m is comparable with n). While it is inevitable that *GS-MC* would be inferior to *GS-Base* in clustering quality, we can expect the sacrifice of quality to be insignificant since documents in a micro-cluster are generally similar to each other in content. Indeed, since we can adjust the size of a micro-cluster (thus also the number of micro-clusters), *MiTexCube* enables flexible efficiency-quality tradeoff.

4.3.2 Query-Specific Cell Summarization

The purpose of query-specific cell summarization is to customize a summary based on the topic preference that a user may have. Specifically, given a set of documents in one text cell as well as a topic keyword query q, the task of query-specific summarization is to generate a summary with P documents selected from the cell that are both representative of the cell and relevant to the query, where P is a number specified by a user to indicate the desired number of documents in the summary. This is different from a traditional information retrieval task, which only considers the relevance of documents to a query. For a summarization task, we also want the selected documents cover well the major content in the cell.

With *MiTexCube*, we can leverage the available micro-clusters to optimize the coverage of the documents in the cell by forcing the summary to include documents distributed over all the distinct micro-clusters. Intuitively, micro-clusters tell us where the redundancy is, because documents within the same micro-cluster are believed to be similar. Specifically, suppose there are K micro-clusters in a cell, given a topic query q we first rank all the documents into a candidate list based on their relevance to the query. Then, in the first round, we select documents from the most relevant one, and if one document is selected, all the documents in the same micro-cluster will be removed from the candidate list and not be considered for selection. The next document to be considered is the most relevant document remained in the list. So this ensure that we select relevant documents distributed over all the micro-clusters. If we need more representative documents (i.e. $P > K$), we just get back all those non-selected documents and do another round of selection.

An *indirect* way to generate a query-specific summary for a text cell is to use a greedy algorithm called Maximal Marginal Relevance (MMR) [10] to avoid redundancy in the selected documents. MMR reranks a list of documents by using the following formula to select the next document to reduce redundancy in the selected documents:

$$argmax_{D_i \in R \setminus S}[\lambda Sim_1(D_i, Q) - (1 - \lambda) \max_{D_j \in S} Sim_2(D_i, D_j)] \tag{4.3}$$

Here, R is the candidate document set, S is the current selected document set, D_i is a candidate document to be considered as the next selected document, Q is a user specified query, Sim_1 is a

function used to measure the similarity between a query and a document, and Sim_2 measures the similarity between two documents.

Compared with the *MiTexCube*-based method (denoted as *QS-MC*), the MMR approach (denoted as *QS-Base*) is less efficient because it requires computation of pair-wise similarity for potentially many document pairs on the fly; besides, the MMR approach may not achieve representativeness well because avoiding redundancy does not always lead to representative topics, while the *MiTexCube* method achieves representativeness more *directly* through the structure based on micro-clusters.

4.3.3 Common Topic Comparison

Another analysis task is to compare multiple text cells to reveal the difference of their coverage on common topics. The standard cell summarization of the text cells cannot easily quantify the coverage of a common topic in different cells, because the result clusters may not be comparable across different cells. A better way to support common topic comparison is to pool the text documents in all the text cells to be compared and cluster them into P clusters, which can then be assumed to be P common topics covered in these cells and serve as a common basis for comparison of different cells. With these P topic clusters as a basis, we can measure the content of each cell by a vector of weights corresponding to the numbers or percentages of documents in the cell that belong to each of the P clusters. Intuitively, such a weight vector (in P-dimensional space) indicates the coverage of each common topic in the corresponding cell, thus comparing these weight vectors across cells can easily reveal which cell covers which topic more and generate trends of topic coverage in any standard dimension with ordinal variables (e.g., location or time).

Once again, *MiTexCube* can speed up this clustering process as we only need to cluster all the micro-clusters in these cells instead of all the documents in them. Without *MiTexCube*, we would have to pool all the documents together and then cluster them from scratch into P common topics. As discussed earlier in the case of standard cell summarization, *MiTexCube* can be potentially much faster than this baseline approach and it also naturally allows us to flexibly take a tradeoff between efficiency and clustering quality.

4.4 Experimental Results

In this section, we will evaluate how well the proposed *MiTexCube* model supports online exploration tasks.

4.4.1 Data Sets

We mainly used the ASRS database [2] for our experiments. We downloaded and extracted two years (1998, 1999) of the data from the database, giving us a total of 4073 records for our experiments. We selected 7 dimensions from the database to construct our *MiTexCube*, and the number of distinct values in each dimension is summarized in Table 4.5.

Table 4.5: Number of distinct values in each dimension of ASRS

State	Flight Condition	Light	Operator	FAR	Flight Phase	Affiliation
3	5	2	8	8	32	10

Another data set we used is the DBLP data set [3]. We totally downloaded 207652 records from the database, which contain papers from year 1989 to 2008 in 75 conferences. Each record has the *title, year, one_author, conference* information about a paper. If one paper has multiple authors, it will generate multiple records with the same *title, year*, and *conference* attributes but different *one_author* attribute. We use the *year, one_author*, and *conference* attributes as the standard fields and use the *title* attribute as the text field. The number of distinct values in each standard dimension of the DBLP data set is shown in Table 4.6.

Table 4.6: Number of distinct values in each dimension of DBLP

Year	Author	Conferece
20	77517	75

Compared with the ASRS data, the DBLP data set has fewer dimensions but much more records. Besides, the number of distinct values in the *author* dimension is much larger than the other two dimensions, while in ASRS data the number of distinct values in each dimension is pretty close to each other. These different characteristics of the two data sets may have different effects on the progressive materialization results. Therefore, we will mainly use the DBLP data set to test the progressive materialization result and compare it with the ASRS data.

4.4.2 Evaluation of Progressive Materialization

We first evaluate the disk storage cost of our progressive materialization strategy. Basically, we will examine the number of concrete cells which are materialized by our strategy as well as their proportion to the total number of cells. There are two important thresholds in our algorithm: the number of micro-clusters K stored in each cell and the upper bound of micro-clusters M that we can deal with efficiently online. We vary both of these parameters to examine how the proportion of concrete cells changes. Besides, we will also examine the scalability of our strategy by varying the number of dimensions and the number of total documents to see how the proportion of concrete cells changes.

In Figure 4.4, we show the results on ASRS data with different number of records used to construct the cube. In Figure 4.4(a) and Figure 4.4(b), we show the comparison among different numbers of micro-clusters stored in each cell (M is fixed to 70), and in Figure 4.4(c) and Figure 4.4(d), we show the comparison of different upper bound M used in our algorithm (K is fixed to 10). We can see that as the number of micro-clusters K stored in each cell increases, both the number and the proportion of concrete cells will increase. On the other hand, the increase of M results in decreasing of the number and the proportion of concrete cells. This is as expected as we have analyzed in previous sections. The overall trend in each of these four figures shows that with the same setting of parameters K and M, the total number of concrete cells will increase as the number of total records goes up.

In Figure 4.5, we show the results on ASRS data with different number of dimensions that we used to construct the cube. We can see that the difference between Figure 4.5 and Figure 4.4 is that as the number of dimensions increases, the proportion of the concrete cells becomes less and less. This is because as the number of dimensions increases, the total number of cells increases much faster than the number of concrete cells. From this perspective, it demonstrates that our progressive strategy scales well with the number of dimensions of the cube.

In Figure 4.6, it shows the results on the DBLP data. Since there are only three dimensions in the data, we only plot the graph as the number of records increases. In both Figure 4.6(a) and Figure 4.6(b), we fix K to 100 and vary M from 200 to 1000. From the results, we find similar trend to the results on ASRS data. But we notice that although the total number of records

in DBLP is 100 times larger than the ASRS data, the proportion of concrete cells in DBLP as shown in Figure 4.6(b) is much smaller than the proportion of concrete cells in ASRS as shown in Figure 4.4(d). This is due to the differences between the characteristics of these two data sets. It indicates that when the number of distinct values among all the dimensions are about the same, then there would be a larger portion of cells need to be materialized than the case where the number of distinct values in each dimension are quite different.

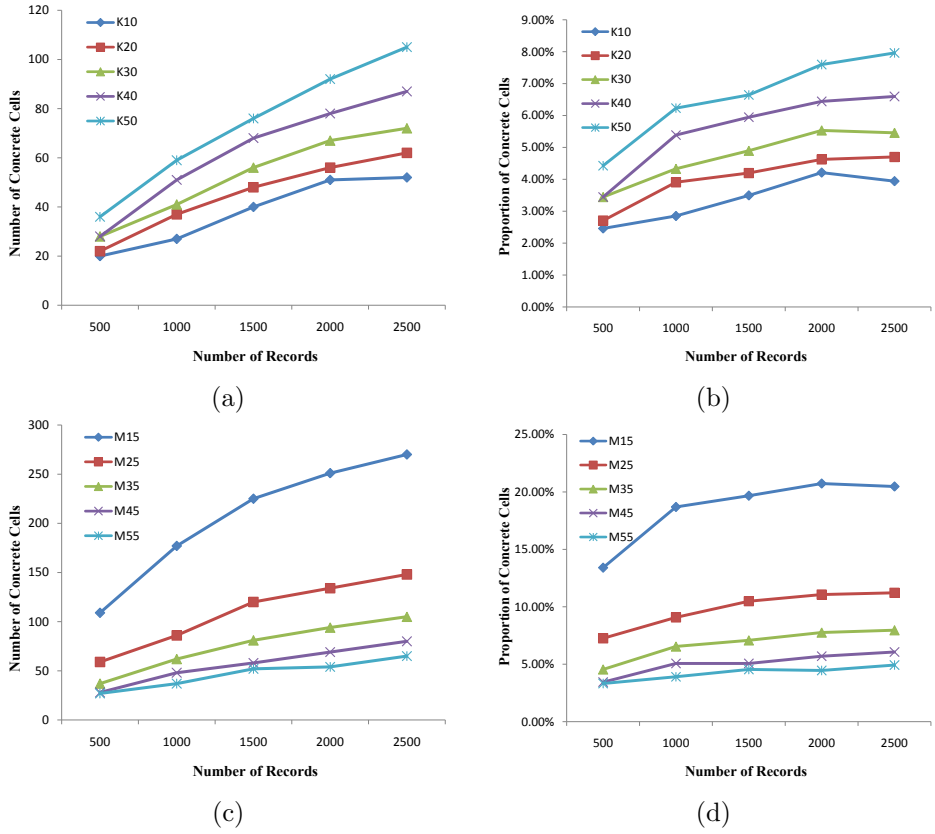


Figure 4.4: Storage Estimation with Different Number of Documents in ASRS

4.4.3 Evaluation of Representative Analysis Tasks

Standard Cell Summarization

We use the ASRS data set to test the standard cell summarization application and compare our *GS-MC* method (based on *MiTexCube*) with the baseline method *GS-Base* (works directly on the documents in a cell), in terms of both efficiency and effectiveness. We vary the parameter K to

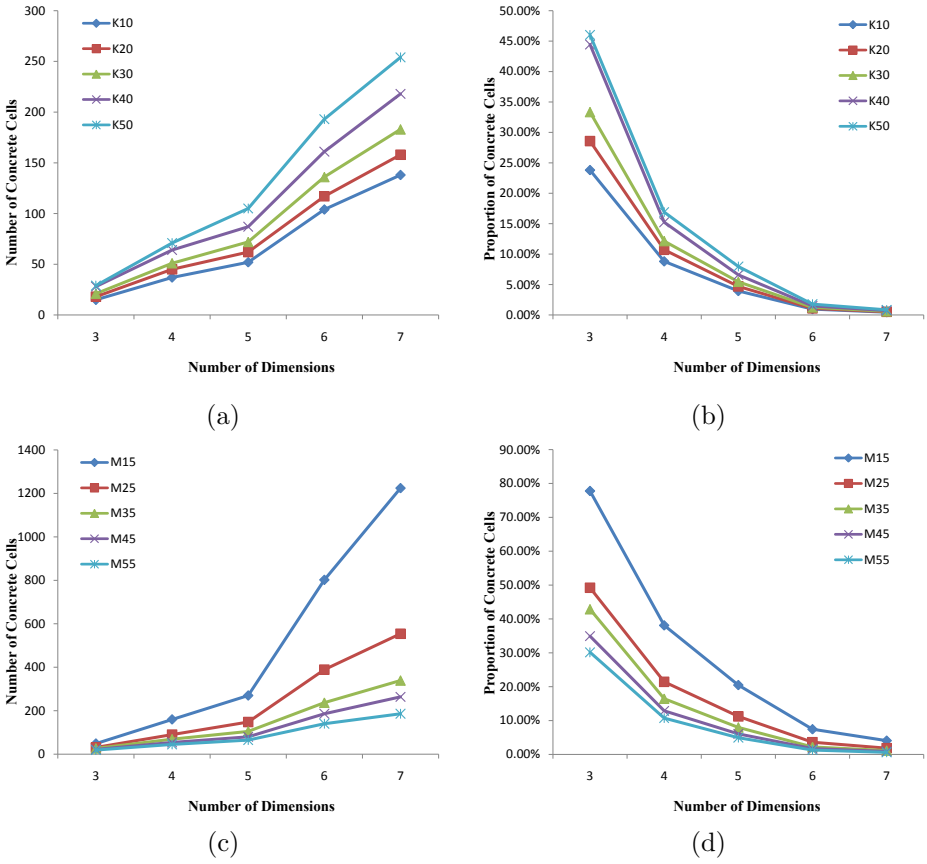
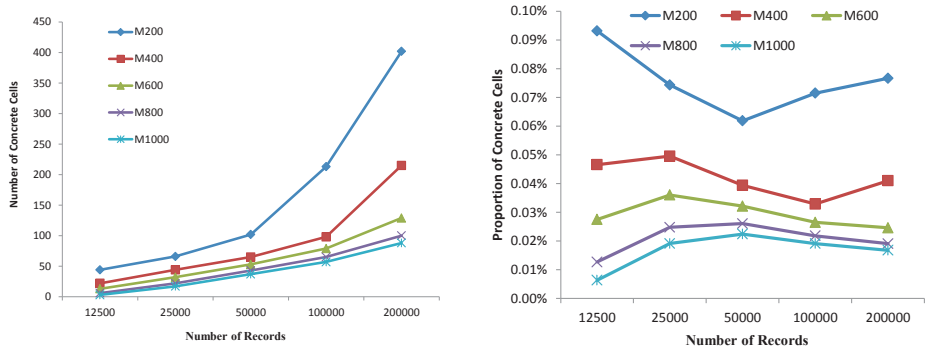


Figure 4.5: Storage Estimation with Different Number of Dimensions in ASRS

generate different settings of *GS-MC* with different numbers of micro-clusters K in each cell, which will be denoted by a suffix indicating the value of K. For example, *GS-MC-100* refers to the setting of $K = 100$.

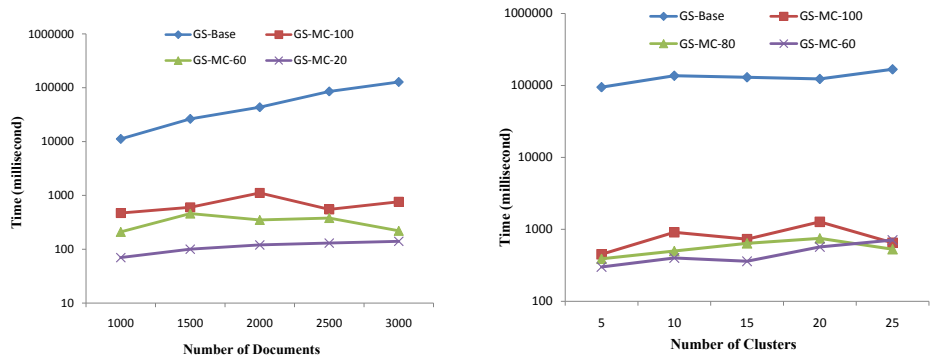
Efficiency: In Figure 4.7(a), we compare the speed of clustering documents or micro-clusters when we vary the size of a cell from 1,000 to 3,000. The target number of clusters in this experiment is 10, which means that we use 10 clusters to summarize the content of documents in each cell. From Figure 4.7, we can see that for all the three different settings (i.e., K=20, 60, and 100), *GS-MC* is much faster than the *GS-Base* method, and for some cells, *GS-MC* is 100 times faster than *GS-Base*. Moreover, as the number of documents increases, the *GS-Base* method slows down dramatically, but the time cost of *GS-MC* does not increase much. In general, the larger the number of micro-clusters K is, the slower the *GS-MC* method is; this is the price we pay for obtaining a finer granularity representation of content, which gives us better approximation of content.



(a) (b)

Figure 4.6: Storage Estimation with Different Number of Documents in DBLP

In Figure 4.7(b), we further compare the two methods by varying the number of targeted clusters. Here, we also test three different settings for *GS-MC*, corresponding to setting K to 60, 80, and 100, respectively. In this experiment, we use a cell which has 2000 documents in it. From the figure, we can make the same conclusion as in Figure 4.7(a), i.e. *GS-MC* is much faster than *GS-Base* in all the settings.



(a) (b)

Figure 4.7: Efficiency Comparison between *GS-Base* and *GS-MC*

Quality of clustering: Since there is always a tradeoff between the efficiency and accuracy, we expect our method *GS-MC* to have inferior quality to the baseline method *GS-Base*, and our main goal is to see how well *GS-MC* can support flexible tradeoff between efficiency and quality. (Indeed, we may view *GS-Base* as a special case of our *GS-MC* when we have each document as a micro-cluster.) Table 4.7 shows the comparison result in a text cell with 2000 documents, and

the number of target cluster P is set to 10 and 5. For each method, we compute its clustering quality as well as its time cost, and the numbers are the average result of 10 runs for each method on each test case. Here, the quality of a clustering result is the sum of cosine similarity between each document vector and its cluster's mean vector, which intuitively captures the coherence of a cluster, and the larger the better.

We tried two strategies to improve the quality of the clustering result.

1. Increasing the number of micro-clusters: During online analysis, when we need finer granularity of micro-clusters to analyze a text cell, we can always go down to its sub cells, which result in a set of larger number of micro-clusters. In our experiment, we test our method with different number of micro-clusters (i.e. K80, K500, K1000), and the result in the table shows that as the number of micro-clusters increase we can get improvement on the quality by sacrificing some time.

2. Additional iterations of document based clustering: After running *GS-MC*, we may also further improve the quality of clustering by starting from the results of *GS-MC* and running additional iterations of k-means on document vectors, as shown in the last six rows of the table where we show the results of running one additional iteration and two additional iterations. For example, K80 + 1 means we do one iteration of document-based clustering after clustering all the 80 micro-clusters into P target clusters, using mean vectors of the result P clusters as the starting point. The result also shows that by additional iterations of document vector based clustering, the quality of clusters can be improved.

Table 4.7: Quality Comparison for Standard Cell Summarization

	P= 10		P= 5	
Method	Quality	Time	Quality	Time
Baseline	491.84	52.36	444.09	47.38
K80	445.59	0.57	408.02	0.50
K500	456.22	6.55	420.82	6.31
K1000	469.87	17.83	430.60	14.86
K80 + 1	463.88	3.53	422.35	2.77
K500 + 1	473.98	9.78	432.84	8.71
K1000 + 1	482.36	21.15	437.90	17.29
K80 + 2	468.11	6.46	427.01	4.98
K500 + 2	477.12	12.97	434.19	11.03
K1000 + 2	484.30	24.42	438.48	19.69

Overall, although the baseline method gets very high quality of cluster, the time cost of it is also the highest. With the help of *MiTexCube*, *GS-MC* can indeed support flexible tradeoff between efficiency and quality of clustering.

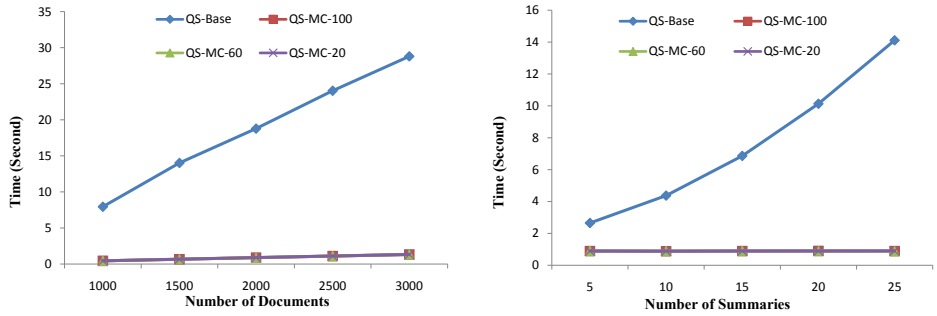
Query-Specific Cell Summarization

We now look at query-specific cell summarization, and here we compare our method *QS-MC* with the MMR baseline method *QS-Base* again in terms of both efficiency and quality. Still, we use the ASRS data as our test data, and use a query ("flight", "system") to test the performance of the two methods (similar conclusions can be drawn with other queries). To calculate the similarity between a document and a query, we use the KL-divergency retrieval model [31].

Efficiency: Figure 4.8(a) and Figure 4.8(b) show the experimental results for different cell sizes and different numbers of summary documents, respectively. From these figures, we can see that the time cost of *QS-Base* increases linearly as we increase either the total number of documents in a cell or the number of target summary documents. If we look at Eq. 4.3, we can find out that in MMR, whenever a top ranked document is selected, it would need to update the score of all the rest documents, and this is the reason why the time cost of *QS-Base* increases linearly as shown in Figure 4.8.

In contrast, the time cost of *QS-MC* only increases very little when the total number of documents increases, as shown in Figure 4.8(a), or when the number of target summary documents increases, as shown in Figure 4.8(b). Moreover, the performance of different settings of the number of micro-clusters K do not have very much difference, so that their curves overlap with each other. Actually, from previous section we can know that the time cost of *QS-MC* mainly depends on the process of document ranking, and almost independent of the number of target summary documents and the setting of the number of micro-clusters K. So overall, the *QS-MC* method is much faster than the *QS-Base* method.

Quality: Table 4.8 shows the quality comparison result of one query when we retrieve 20 document as a summary based on two measures: (1) coverage and (2) relevance. The coverage is calculated using the following method: for each unselected document, we calculate the highest cosine similarity of this document with the selected 20 documents as its score, which intuitively captures how well



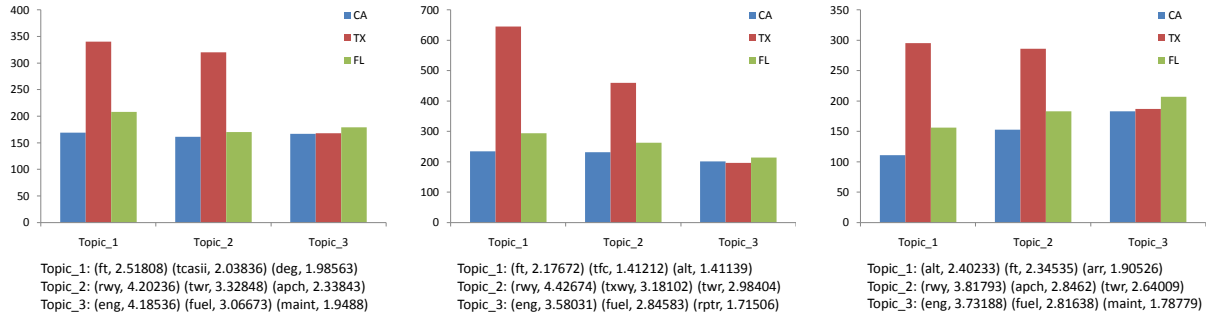
(a) (b)

Figure 4.8: Efficiency Comparison between *QS-Base* and *QS-MC*

this document is covered by the selected 20 documents. Then, we sum over the scores of all the unselected documents as the coverage. Relevance is the total similarity of all the selected document to a query. The top three rows are results of MMR (i.e., *QS-Base*), and the bottom three rows are result of our method (i.e., *QS-MC*), where K is the number of micro-clusters in the cell and λ is the weight parameter used in MMR. The total number of document is 2000. From these results, we can see that *QS-MC* consistently outperforms *QS-Base* in coverage due to better capturing the representative topics in the cell through micro-clusters, confirming our hypothesis that direct modeling topics through micro clusters is more effective for selecting documents representing the cell well than the indirect way through eliminating redundancy used in MMR. However, we also note that *QS-MC* has lower relevance than *QS-Base*, which indicates a tradeoff between relevance and coverage as well as a tradeoff between relevance and efficiency (as discussed earlier, *QS-Base* is much less efficient than *QS-MC*). Note that here again *MiTexCube* allows us to make flexible tradeoff between relevance and efficiency since as K get larger, we get better relevance.

Table 4.8: Quality Comparison for Topic-biased Cell Summarization

λ	0.2	0.4	0.6	0.8	1
relevance	-283.27	-282.278	-282.278	-282.218	-282.21
coverage	258.28	257.919	257.919	259.707	259.707
K	10	20	30	50	100
relevance	-284.86	-284.0996	-282.8749	-282.6589	-282.5442
coverage	264.3687	269.9924	271.238	264.84	267.6433



(a) (b) (c)

Figure 4.9: Common Topic Comparison

Sample results of comparative analysis of text cells

We use sample results from the ASRS data to show the effectiveness of *MiTexCube*in the *Common Topic Comparison* task. We use the total 4071 documents within three cells for the comparison, which have different locations(states), namely CA, TX, and FL. The number of common topics to be compared is set to 10.

Figure 4.9(a) shows the comparison result based on document units[1]. Figure 4.9(b) is the result based on micro-clusters in which each cell has 100 micro-clusters inside, and Figure 4.9(c) is the result where each cell has 500 micro-clusters. The y-axis is the number of documents that belong to one topic within a cell. The three topics on the x-axis are the top three major topics within the 10 common topics. The top weighted terms are also listed under each graph. We can see that the two micro-cluster based methods got similar comparison result to the document unit based method. When the number of micro-clusters of each cell increases, the comparison results are much closer to the document unit based approach. For example, for the comparison of topic_3 over different states, Figure 4.9(c) is more accurate than Figure 4.9(b). In addition, compared with the K100 based approach, the K500 based approach has more similar top weighted terms to the document based approach. The time cost for these three methods are: 215.77, 18.09, and 62.35 seconds, which shows the advantage of micro-cluster based methods in terms of efficiency.

[1]Abbreviations: (ft: Feet), (tcasii: Traffic Alert and Collision Avoidance System), (deg: Degree), (rwy: Runway), (twr: Tower), (apch: Approach), (eng: Engine), (maint: Maintenance), (tfc: Traffic), (alt: Altitude), (txwy, Taxiway), (rptr: Reporter)

4.5 Conclusions and Future Work

In this Chapter, we proposed a novel cube called MicroTextCluster Cube (*MiTexCube*) to enable efficient online analysis of text cells in several applications. We proposed a progressive materialization algorithm for this novel cube, an update algorithm for the cube when there are changes in the database, and methods to leverage *MiTexCube* for three analysis tasks, including standard cell summarization, query-specific cell summarization, and common topic coverage comparison. Experimental results on real multidimensional text databases show that applications based on the proposed materialized *MiTexCube* are more efficient than the baseline methods of direct analysis based on document units in each cell, without sacrificing much quality of analysis. The proposed *MiTexCube* has several parameters to accommodate flexible tradeoffs between time and space as well as effectiveness and efficiency. We conducted several experiments to understand the effects of changing these parameters and discussed how to set these parameters empirically.

Chapter 5

Probabilistic Topic Mapping Model for Mining Parallel Text Fields

In the last two chapters, we discussed *Topic Cube* and *MiTexCube*, and their usage in digesting and exploring an MDT database. To better take the advantage of the rich information within the database, *e.g.* to support decision making, it is often necessary to go beyond supporting digestion and exploration to further support analysis of latent patterns within the database. To achieve this goal, in this chapter I will describe a novel model for discovering and comparing latent topical patterns embedded in text data. The proposed model performs comparative analysis of two parallel text fields in an MDT database to extract topics from each field and discover their associations. This kind of MDT databases, *i.e.* MDT databases with multiple text fields, exist in many domains. For example, in IT service domain, an MDT database has both "Problem" and "Solution" text fields, and in medical care domain, an MDT database always has both "Symptom" and "Treatment" text fields. A record in such a database always has text information in both of these two text fields, which means a document in one text field is always accompanied by a document in another text field. Comparative analysis on these text dimensions could discover many useful knowledge. For example, by analyzing the correlations between the "Problem" field and the "Solution" field, an IT service manager could figure out possible solutions to a certain type of IT service problems, and this kind of knowledge is very desirable. Therefore, in this chapter, we are interested in mining two text fields in an MDT database, which we also call a *parallel document collection*. In general, a parallel document collection consists of two sets of documents *source* and *target* such that a document of source set is associated with a document of target set. For mining such a collection, I proposed and studied a novel model called Probabilistic Topic Mapping (PTM). Experimental results show that PTM can not only mine meaningful knowledge out of a parallel collection, but is also capable of mining millions of records with a Hadoop implementation.

5.1 Introduction

We present two motivating examples to illustrate real parallel document collections in MDT databases and the need for mining them.

Help Desk Support: Service providers such as help desks are continuously seeking improved techniques to diagnose problems, identify their root cause and develop and deploy solutions quickly. Typically, service providers deploy workflow systems that track the lifecycle of problem records from occurrence to resolution. Over time, a large number of these records are collected by the provider, which constitutes the core knowledge-base for IT problems and solutions. Such records naturally form a parallel document collection where one set of documents includes all the problem records, the other has all the solution records, and each problem record and solution record form a semantic pair. For example, Table 5.1 shows an example problem record from a service provider. Such problem-solution data contains valuable knowledge about problem solving strategies as well as problem patterns, making it interesting to study how to mine such data to discover useful knowledge. One particularly interesting goal is to discover the major problem areas, major solution strategies and what solution strategies can be applied to what problem areas. That is, to discover latent topics from both problem set and solution set, and to map problem topics to solution topics.

Disease Diagnosis: As another example, consider the medical text data shown in Table 5.2, where there are two separated text fields for each kind of disease. We can view all the text under "symptoms" and "treatments" as two separate sets of documents that can be paired with each other. Although the vocabularies in symptom set and treatment set are often very different, the semantic topics and their mapping from symptom to treatment are often very consistent. For such a parallel document collection, it would be extremely valuable to discover the major topics on symptoms and treatments, and further understand the mapping between symptoms and treatments so that we can map a category of symptoms to potentially multiple categories of treatments.

In both examples, we see that the general mining problem is as follows: Given a parallel document collection, discover a set of topics from each set of documents and map a topic from one set to potentially multiple topics in the other. Although much work has been done on text mining (see Section 2 for a detailed review of related work), no previous work has addressed this novel problem. The challenges for mining parallel document collections are the followings:

- How to simultaneously mine two sets of topics and their correlations? One challenge here is the vocabulary gap existent between source and target document sets. E.g., in the medical example, although a symptom document companies with a treatment document, the vocabularies are often very different. So it's not feasible to first mine two sets of topics and then match them based on the content of the topics.

- How to handle the different granularities of topics in source and target sets? E.g., in the medical example, similar symptoms can lead to very different treatments. It is important to allow flexible mapping between source and target topics.

- How to construct the model over millions or billions of parallel documents? E.g., there are tons of IT service tickets generated everyday, and mining such a large scale of data is highly desirable.

- How to design a model which can flexibly incorporate prior belief? E.g., in certain analysis, symptom and treatment topics often need to be calibrated with physician's own belief in order to generate clinically meaningful topics.

To address all these challenges, we propose to solve this mining problem with a novel probabilistic topic model and a scalable parallel EM implementation. Specifically, we propose a new probabilistic topic model, called Probabilistic Topic Mapping (PTM) model, to mine parallel document collections to simultaneously discover latent topics in both collections as well as the mapping of topics in one collection to those in the other. We evaluate the PTM model on two different parallel document collections in two representative domains. One is from IT service management domain, which we collected from a commercial IT service problem management system; the other is a medical domain symptom-treatment data set derived from Google Health [1]. Evaluation results show that PTM can effectively discover meaningful topics and mappings on both data sets. We also show that the discovered topic mappings can be used to improve text matching when there's a vocabulary gap. We further implement PTM with MapReduce on a large Hadoop cluster and test its scalability. The results show that PTM can scale up to parallel document collections with million documents on a large Hadoop cluster.

[1] https://health.google.com/health/ref

Table 5.1: Examples of Ticket Data

Topic	Problem	Solution
Capacity	Received TEC alert: Disk space C: increased to 91%, only 443MB is free.	Deleted temp files to free up space
Backup	Investigate why server shows failed or incomplete on the backup report	Verified backup service on servers
Printer	Please create a print queue for our site...	cleared print queue and print test page
Hardware	USSBHDD1- Disk failure	mirrored disk was replaced

Table 5.2: Examples of Medical Data

Disease	Symptom	Treatment
Food allergy	Symptoms usually begin immediately, within 2 hours after eating. Rarely, the symptoms may begin hours after eating the offending food...	The only proven treatment for a food allergy is to avoid the food. If you suspect you or your child has a food allergy, consult an allergy specialist...
Sciatica	Sciatica pain can vary widely. It may feel like a mild tingling, dull ache, or a burning sensation. In some cases...	Treatment is aimed at maximizing mobility and independence. The cause of the nerve dysfunction should be identified and treated as appropriate. In some cases...

5.2 Problem Formulation

Our input consists of two sets of documents from two text fields of an MDT database, where documents in one set are paired with those in the other. We would assume that one set of documents to be *source documents* and the other to be *target documents*, and the corresponding sets would be referred to as *source document set* and *target document set*, respectively. Our goal is to (1) discover the major topics from both the source set and target set, and (2) discover a mapping from source topics to target topics.

We now define this problem more formally.

Definition 5.2.1 (Parallel Document Collection) *A Parallel Document Collection C is a set of text document pairs, i.e., $C = \{(s_1, t_1), (s_2, t_2), \ldots, (s_N, t_N)\}$, where s_i and t_i are two text documents and are referred to as a* source document *and* target document*, respectively; N is the total number of document pairs in the collection. We also refer to (s_i, t_i) as a* parallel document*, and denote it by d_i.*

Given a parallel document collection D, we also refer to the set of all source documents as the

source set, $C_s = \{s_1, s_2, \ldots, s_N\}$, and similarly, the set of all target documents as the *target set*, $C_t = \{t_1, t_2, \ldots, t_N\}$. Thus, we may also regard the entire parallel document collection C as $\{C_s, C_t\}$. For example, Table 5.1 shows a set of parallel documents, each of which contains a Problem field (source document) and a Solution field (target document).

Definition 5.2.2 (Topic) *A topic in a text collection (either a source set or a target set) is a probabilistic distribution over words, which characterizes a semantically coherent topic in the collection. Formally, a topic is represented by a unigram language model θ, i.e. a word distribution $\{P(w|\theta)\}_{w \in V}$ s.t. $\sum_{w \in V} P(w|\theta) = 1$. Here, V denotes the whole vocabulary of our corpus.*

A word with high probability in such a distribution often suggests what the topic is about. For example, a probability distribution which has high probability over the words "tape", "restore", and "incomplete" may suggest a topic about backup of data. Such a definition of topic has been commonly adopted in most of the existing work on using topic models for text mining (e.g., [27, 9]).

Definition 5.2.3 (Mining Topic Mapping (MTM)) *Given a parallel corpus $C = \{C_s, C_t\}$, the goal of mining topic mapping (MTM) from C is to mine K_s topics $\{\theta_i\}_{i=1}^{K_s}$ from the source set C_s and K_t topics $\{\psi_j\}_{j=1}^{K_t}$ from the target set C_t, as well as the $K_s \times K_t$ topic mapping probabilities from the source to the target document set, i.e. $P(\psi_j|\theta_i)$ for $i = 1, \ldots, K_s$ and $j = 1, \ldots, K_t$. See Figure 5.1 for a visual illustration of a possible output of MTM.*

A unique novelty of the MTM task is that it not only extracts two sets of topics from two correlated text document sets, respectively, and but also discovers mappings between these two sets of topics, which no previous work has attempted to discover.

For example, in the ticket data described in Table 5.1, MTM will mine topics like "Capacity Problem" and "Hardware Problem" from the source set (i.e., problem set) and topics like "Deletion Operation" and "Replace Operation" from the target set (i.e., solution set). At the same time, the task will also indicate which solution topic is the most appropriate one for a problem topic. For example, the "Deletion Operation" could be the best solution for the "Capacity Problem". An example of the mining result is shown in Figure 5.1.

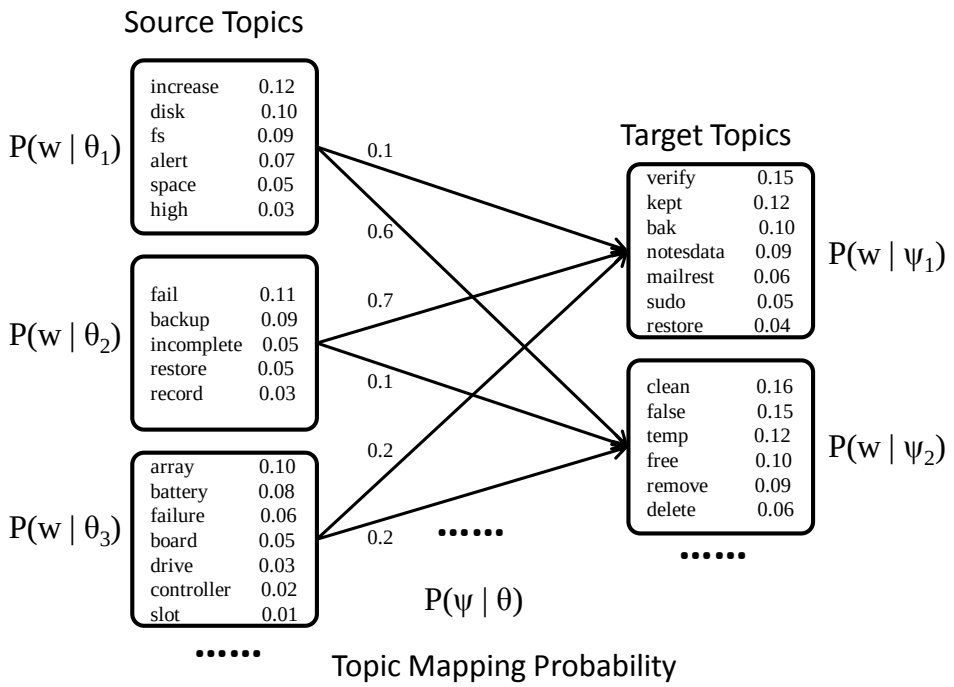


Figure 5.1: An Illustration of Mining Topic Mapping

5.2.1 Generating a Parallel Document Collection from an MDT Database

For an MDT database with multiple text fields, it naturally contains a parallel document collection, *e.g.* the MDT databases shown in Table 5.1 and Table 5.2. For MDT databases with only one text field, although there is no natural parallel document collections available, it's still possible to generate a parallel document collection from the database and use PTM to mine useful knowledge out of it. The idea is to first partition all the records in the database into two groups and then associate documents from different groups by standard dimension values. For example, in the MDT database shown in Table 1.1, we can first partition all the records by "Daylight" and "Night", and then associate the narratives from "Daylight" and "Night" by the same date or the same flight model to generate a parallel document collection. In this way, any MDT databases can generate several possible parallel document collections, and then PTM can be used for comparative analysis on the collections.

5.3 Probabilistic Topic Mapping

In this section, we present a novel probabilistic topic model called Probabilistic Topic Mapping (PTM) model to solve the problem of mining topic mappings from a parallel document collection. We will first define the model and then discuss how to estimate the parameters of the model.

5.3.1 Model Description

The idea of PTM is to introduce two sets of word distributions to model the topics in a source set and a target set, respectively, use a set of possible topic mapping probabilities to model the topic mapping relation, and then assume a parallel document collection is a sample drawn from a mixture model involving these word distributions and topic mapping probabilities. Therefore, given a parallel document collection, we can then fit the model to the data and estimate all those parameters. The estimated parameter values would give us the specific topics in each set and specific mapping probabilities between topics.

Formally, let $\{\theta_i\}_{i=1}^{K_s}$ be K_s topics in the source set C_s and $\{\psi_j\}_{j=1}^{K_t}$ be K_t topics in the target set C_t, where $p(w|\theta_i)$ and $p(w|\psi_j)$ are word distributions for each topic θ_i and ψ_j. Let $P(\psi_j|\theta_i)$ ($i = 1, \ldots, K_s$ and $j = 1, \ldots, K_t$) denote the probability that topic θ_i of the source set should be mapped to topic ψ_j in the target set. These probabilities essentially encode the major knowledge we would like to discover from a parallel document collection.

Now we assume that a parallel document $d = (d_s, d_t)$ (d_s and d_t are the source and target documents) is generated word by word in the following way:

1. To generate a word w in the source document d_s:
   (a) Pick a topic $z_s = \theta_i$ with probability $P(\theta_i|d)$
   (b) Sample a word w from the multinomial distribution $P(w|\theta_i)$
2. To generate a word w in the target document d_t:
   (a) Pick a topic from the source set, $z_s = \theta_j$ with probability $P(\theta_j|d)$
   (b) Pick a topic in the target set, ψ_i, according to the topic mapping probability distribution $P(\psi_i|\theta_j)$

(c) Generate a word w from the multinomial distribution $P(w|\psi_i)$

Note that we model the source document and the target document differently to capture the asymmetric mapping from topics in the source set to those in the target set. The mapping is reflected in step 2.(b) when generating a target document, where we first sample a topic from the source set and use the topic mapping model $p(\psi_i|\theta_j)$ to select a topic in the target set. This ensures the coupling of the topic choices for the source and target documents, and the discovered topics on the two sets would be aligned with each other. The graphical representation of PTM is described in Figure 5.2. Note that no previous model outputs topics as well as their mappings as we do, and if we apply topic models to each set separately, the mined topics from the two sets may not be well-aligned with each other.

Based on the generative process described above, every word in the source document of d can be potentially generated using any of the K_s topics, thus the probability of generating word w is:

$$P_s(w|d) = \sum_{j=1}^{K_s} P(\theta_j|d)P(w|\theta_j), \tag{5.1}$$

To compute the probability of generating a word w in the target document, we first need to compute the probability that a topic ψ_i would be selected, i.e., $p(\psi_i|d)$. Since ψ_i is selected through a topic mapping probability distribution $P(\psi_i|\theta_j)$ and θ_j can be any of the K_s source topics, we have

$$P(\psi_i|d) = \sum_{j=1}^{K_s} P(\psi_i, \theta_j|d) = \sum_{j=1}^{K_s} \{P(\psi_i|\theta_j)P(\theta_j|d)\} \tag{5.2}$$

Once $p(\psi_i|d)$ is known, we can compute the probability of generating word w in the target document of d in the same way as we did for a word in the source document. That is,

$$P_t(w|d) = \sum_{i=1}^{K_t} P(\psi_i|d)P(w|\psi_i) \tag{5.3}$$

$$= \sum_{i=1}^{K_t} \left\{ \sum_{j=1}^{K_s} P(\psi_i|\theta_j)P(\theta_j|d) \right\} P(w|\psi_i) \tag{5.4}$$

Therefore, given the document IDs for all the documents in a parallel document collection, the

log-likelihood of the whole parallel document collection C is

$$\begin{aligned} L(C) &= \sum_{d\in C}\sum_{w\in V}\{c(w,d_s)\log P_s(w|d) + c(w,d_t)\log P_t(w|d)\} \\ &= \sum_{d\in C}\sum_{w\in V} c(w,d_s)\log\sum_{j=1}^{K_s} P(w|\theta_j)P(\theta_j|d) \\ &\quad + \sum_{d\in C}\sum_{w\in V} c(w,d_t)\log\sum_{i=1}^{K_t}\left\{\sum_{j=1}^{K_s} P(\psi_i|\theta_j)P(\theta_j|d)\right\}P(w|\psi_i), \end{aligned} \tag{5.5}$$

where $c(w,d_s)$ and $c(w,d_t)$ are the count of w in d's source document d_s and target document d_t, respectively. The proposed PTM model has the following parameters, which we will denote

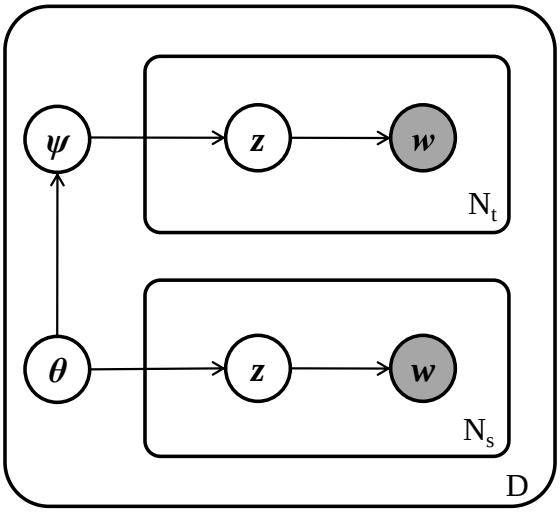


Figure 5.2: Graphical Model for PTM

by Λ: (1) source topics: $\{P(w|\theta_i)\}_{V\times K_s}$; (2) target topics: $\{P(w|\psi_j)\}_{V\times K_t}$; (3) topic mapping: $\{P(\psi_j|\theta_i)\}_{K_t\times K_s}$; and (4) coverage of source topics in each parallel document d: $\{P(\theta_i|d)\}_{K_s\times N}$. They satisfy the following constraints:

$$\begin{cases} \sum_{i=1}^{K_s} P(\theta_i|d) = 1 & \textit{for } d\in C \\ \sum_w P(w|\theta_j) = 1 & \textit{for } j = 1,\ldots,K_s \\ \sum_w P(w|\psi_i) = 1 & \textit{for } i = 1,\ldots,K_t \\ \sum_{i=1}^{K_t} P(\psi_i|\theta_j) = 1 & \textit{for } j = 1,\ldots,K_s \end{cases}$$

We can now see that the problem of mining topic mappings is now reduced to one to compute the values of these parameters based on a given parallel document collection. In the next section, we will discuss how we estimate these parameters so that we can obtain the mined topics in both the source and document sets and the mappings of topics from source to target.

5.3.2 Parameter Estimation

In this section, we discuss how to estimate the parameters Λ using the maximum likelihood estimator, which selects parameter values that maximize the data likelihood. That is, our estimate would be given by

$$\Lambda^* = \arg max_{\Lambda} L(C|\Lambda)$$

where $L(C|\Lambda)$ is the log-likelihood of the parallel document collection C given in Equation 5.5 (here we introduced Λ into the function to make the optimization problem more explicit).

Since PTM is a mixture model, there is no analytical solution to the maximum estimation problem. However, as in the case of other mixture models, we can use use the Expectation-Maximization (EM) algorithm to estimate the parameters. The EM algorithm is an iterative hill-climbing algorithm. It starts with a randomly chosen initial value for Λ and then iteratively improves it until it reaches a local maximum of the likelihood function. In each iteration, it would update the parameters through an E-Step and an M-step.

In the E-step, we use the current version of parameter values to infer the latent topics sampled in the generation process (i.e., the z_s and z_t described in the generation process) by computing the probability distributions of three hidden variables: (1) $\{z_{d_s,w} \in [1, K_s]\}$ indicates the source topic used to generate word w in source document d_s (thus $P(z_{d_s,w} = j)$ gives the probability that word w in d_s is generated from source topic θ_j). (2) $\{z_{d_t,w} \in [1, K_t]\}$ indicates the target topic used to generate word w in a target document d_t. (3) $\{z_{d_t,\psi_i} \in [1, K_s]\}$ indicates the *source* topic used to generate the target topic ψ_i through mapping, thus $P(z_{d_t,\psi_i} = j)$ is the probability that the target topic ψ_i in d_t is selected based on (i.e., mapped from) the source topic θ_j. The updating formulas for the E-step are shown in Eq. 5.6 to Eq. 5.8, where $P^{(m)}$ indicates the value of a parameter

estimated in the *m-th* step of the EM algorithm.

$$P(z_{d_s,w} = j) \propto P^{(m)}(w|\theta_j)P^{(m)}(\theta_j|d) \tag{5.6}$$

$$P(z_{d_t,w} = i) \propto P^{(m)}(w|\psi_i)\sum_{j=1}^{K_s} P^{(m)}(\psi_i|\theta_j)P^{(m)}(\theta_j|d) \tag{5.7}$$

$$P(z_{d_t,\psi_i} = j) \propto P^{(m)}(\psi_i|\theta_j)P^{(m)}(\theta_j|d) \tag{5.8}$$

In the M-step, we use the hidden variable distributions obtained from the E-step to re-estimate all the parameters. Specifically, the formulas for updating all the parameters of our PTM model are as shown in Eq. 5.9 to Eq. 5.12, where all the parameters would be normalized with their corresponding constraints so that all the probabilities would sum to 1.0. Basically, we just pool together the expected counts for all these parameters with respect to the distributions of the hidden variables computed in the E-step.

$$P^{(m+1)}(w|\theta_j) \propto \sum_d c(w, d_s)P(z_{d_s,w} = j) \tag{5.9}$$

$$P^{(m+1)}(w|\psi_i) \propto \sum_d c(w, d_t)P(z_{d_t,w} = i) \tag{5.10}$$

$$P^{(m+1)}(\psi_i|\theta_j) \propto \sum_d \sum_w c(w, d_t)P(z_{d_t,w} = i)P(z_{d_t,\psi_i} = j) \tag{5.11}$$

$$P^{(m+1)}(\theta_j|d) \propto \sum_w c(w, d_s)P(z_{d_s,w} = j) + \sum_w c(w, d_t)\left\{\sum_{i=1}^{K_t} P(z_{d_t,w} = i)P(z_{d_t,\psi_i} = j)\right\} \tag{5.12}$$

The EM algorithm is only guaranteed to converge to a local maximum, so in general, we run multiple trials and select the best local maximum as an approximation of the true global maximum.

5.3.3 Incorporating Prior Knowledge into PTM

Sometimes we have some prior knowledge about the topics or topic mappings in an application domain that we may wish to leverage in mining topic mappings. For example, an analyst may have some knowledge about topics exist in the collection and so can easily specific some keywords to define a topic (e.g., words like "print", "ink", and "queue" may be provided for a topic about printing problems). Also, we may know that the mapping from source topics to target topics is

one-to-one. We now discuss how we can incorporate such prior knowledge into PTM and obtain interesting variants of the "standard" PTM introduced earlier.

First, instead of using the maximum likelihood estimator to estimate the parameters of PTM, we can use our prior knowledge to define a prior distribution of parameters and use the Maximum A Posteriori (MAP) estimator to estimate the parameters. The MAP estimator would attempt to maximize both the likelihood and the consistency with the defined prior. If we define a conjugate prior to our model, the MAP estimator can be computed using the same EM-algorithm as we used for the Maximum Likelihood estimator except that the M-step will involve pooling additional "pseudo counts" from the prior with the expected counts in the original formulas. In this sense, a conjugate prior thus in effect would convert our prior knowledge into additional "pseudo data" to improve our estimate of parameters. For example, when we describe a topic about printing problem with words such as "print" and "queue", with a conjugate prior, we would pretend we had observed additional counts of these words in the M-step, leading to higher probabilities for these words, and thus effectively "forcing" the estimated topic to be closer to our topic specification. Such a way of allowing a user to control a topic model has also been used in previous work [49].

Second, we may also use our prior knowledge to set some parameters of the PTM model. Here we discuss one interesting case, where we know that the topic mapping is one-to-one. In this case, the numbers of source and target topics would be identical, i.e., $K_s = K_t$. Moreover, a source topic can only be mapped to precisely one target topic, so we can assume that the topic choice in both the source document and the target document to be identical, i.e., $P(\theta_i|d) = P(\psi_i|d),\ for\, i = 1, \ldots, K_s$. Intuitively, such a model would allow us to *align* each topic in the source documents to precisely one topic in the target documents and vice versa. Thus we call such a special case of PTM an Alignment PTM (APTM), and its graphical illustration is shown in Figure 5.3.

The estimation of the APTM can be done by using the same EM algorithm used for the standard PTM except that we do not need to estimate $p(\psi_i|\theta_j)$ as it is known that $p(\psi_i|\theta_i) = 1$, while $p(\psi_i|\theta_j) = 0$ if $i \neq j$.

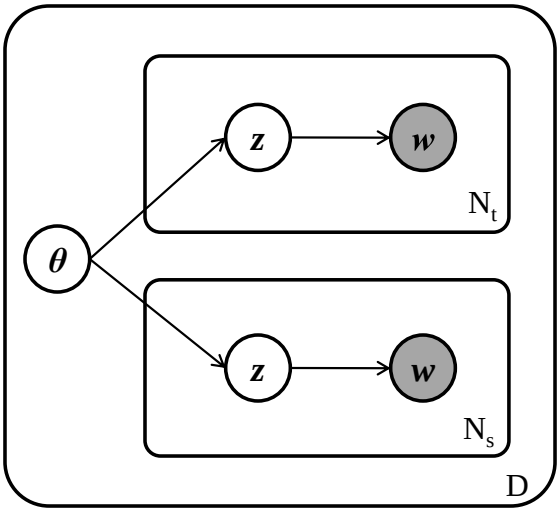


Figure 5.3: Graphical Model for APTM

5.4 Experiments

In this section, we evaluate our PTM model both qualitatively and quantitatively. Besides, we also test the scalability of our PTM model using the Hadoop framework, which demonstrates the feasibility of PTM in real business in which at least millions of parallel documents need to be analyzed.

5.4.1 Data Set

We used two data sets from two different domains, namely IT service domain and medical domain, to evaluate the proposed PTM model. The two data sets also represent two different ways to form a parallel document collection.

The first data set is collected from a commercial IT service problem management system. Everyday, thousands of tickets are delivered to this IT service problem management system. Once a problem ticket is resolved, the agent will document the solution for future references. All these archived data need to be analyzed for further improving the whole IT services for their customers. Our PTM model can serve as a powerful tool for analyzing this data, where we treat each ticket's problem and solution as a parallel document pair. For different experiment purposes, we use subsets of different sizes collected from this large data archive.

Table 5.3: Word distributions of topics mined from PTM in ticket data

Capacity		Hardware	
Problem	Solution	Problem	Solution
increase 0.06	**free** 0.04	array 0.08	**replace** 0.07
space 0.05	**temp** 0.04	battery 0.06	**battery** 0.04
disk 0.04	**delete** 0.04	accelerate 0.05	**cache** 0.03
alert 0.04	**clean** 0.03	failure 0.03	**ce** 0.02
filesy 0.03	**file** 0.03	drive 0.03	**adu** 0.02
cpu 0.03	**false** 0.02	fail 0.03	**array** 0.02
tec 0.03	**normal** 0.02	controller 0.02	**ok** 0.02
diskused 0.03	**fs** 0.02	slot 0.02	**raid** 0.02
91pct 0.03	**usage** 0.02	board 0.02	**detect** 0.01
high 0.03	**old** 0.02	attach 0.02	**fine** 0.01

The second data set is collected from medical domain and is downloaded from Google Health. For each disease, we use the description of its symptom and treatment as the parallel document pair. Totally, we collected 1300 pairs of parallel documents from this web site.

5.4.2 Sample Topic Mapping Results

The output from our PTM model consists of (1) a set of source topics; (2) a set of target topics; and (3) mapping relations of the source topics and target topics. We now show some sample mining results for all these three components of the mining results. We first show two sample source topics and their corresponding target topics mined from the problem-solution ticket domain in Table 5.3. For each topic, we show the top k words with the highest probability in the word distribution of that topic (i.e., $p(w|\theta)$ or $p(w|\psi)$). The two columns labeled as "*Capacity Problem* and *Capacity Solution*" are a sample pair of source topic and target topic with high mapping probability. We see that they are intuitively very meaningful, showing that disk space is a major problem in this data set and its common solution is to delete temporary files and free up disk space. Similarly, the two columns labeled as "*Hardware Problem* and *Hardware Solution*" are another pair of source topic and target topic with high mapping probability. Once again, we see that the mined topics and their mapping are very meaningful, indicating that another major category of problems is hardware problems. Clearly, such PTM results would be very useful to help analysts understand the major problems and their solution strategies.

We show some sample topic mapping results with PTM on our medical data set in Figure 5.4 and Figure 5.5. Figure 5.4 shows the result where we set more target topics ($K_t = 40$) than source topics ($K_s = 20$), while in Figure 5.5 the number of source topics is larger. In the first case, one symptom can have several different treatments. For example, the ("abdominal", "vomit", "diarrhea") symptom could be either diagnosed as "allergy" to some "food" and treated with "antihistamine", or be diagnosed as "intestine" or "bowel" disease and treated by "vitamin" and "nutrition" "supplement". The modeling result of PTM also provides the possibilities of these two treatments according to the data. In the second case, when the number of symptom topics is larger than the number of treatment topics, we can find some treatments which can be used for several different symptoms. For example, the "antibiotic" and "penicillin" treatment can be used for both ("fever", "headache") symptom and ("ulcer", "blister") symptom. In summary, from this example, we can see that with different settings, PTM can reveal different correlations between topics in two parallel document collections, providing the needed flexibility for an analyst to probe a data set according to different application needs.

In contrast, the sample results in Figure 5.6 show that APTM only allows one to one topic mappings. So if one symptom has several possible treatments, the corresponding treatment topic of that symptom topic may mix several treatments together. For example, Figure 5.6 shows the mining result with APTM on the same medical data. On the left, "Symptom_1" has almost the same content as the "Symptom_1" in Figure 5.4. While in Figure 5.4 we know there are several possible treatments for this symptom and their possibilities, in Figure 5.6 all the possible treatments are mixed together. This is not very clear for an analyst to analyze the correlations between topics embedded in the data. A similar example is shown on the right in Figure 5.6, where one treatment can resolve several symptoms and all the those symptoms are mixed together in the mining result of APTM.

In both domains, we see that the PTM model can discover meaningful topics from both the source set and the target set and map source topics to target topics to reveal interesting associations.

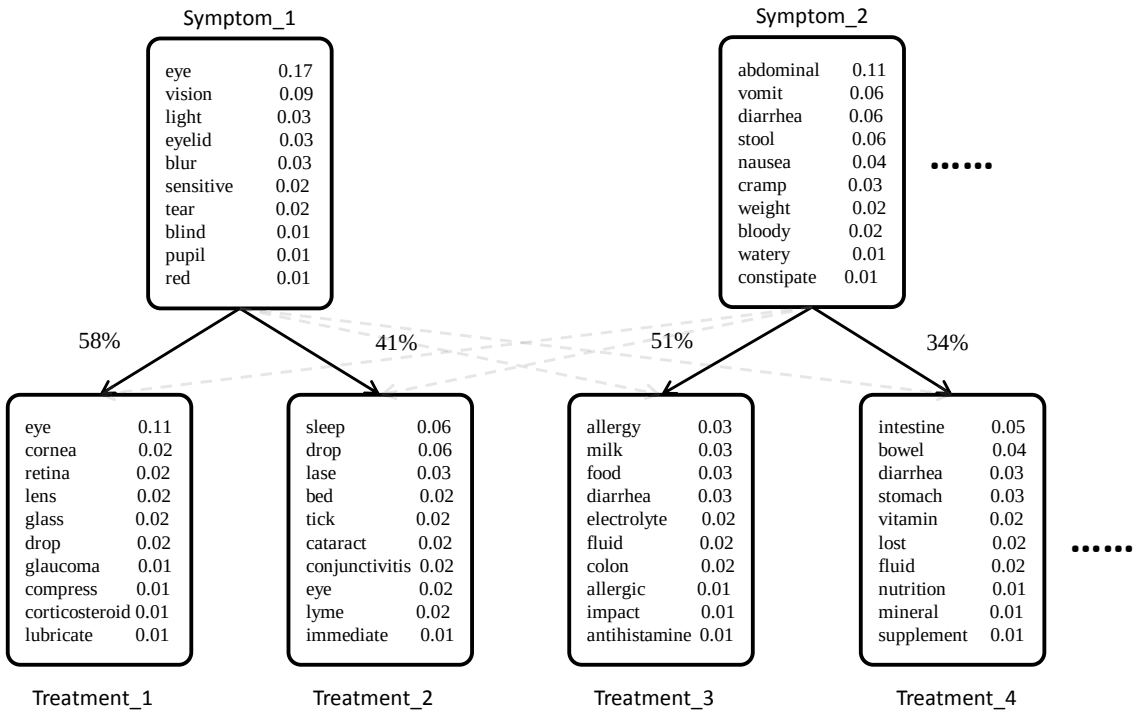


Figure 5.4: Word distributions and topic mapping learned from PTM on medical data ($K_s = 20, K_t = 40$)

5.4.3 Quantitative Evaluation of PTM

To quantitatively evaluate the quality of the discovered topics and their mappings by PTM, we study an interesting application of PTM for matching a document in the source set with documents in the target set. That is, given a document in the source set, we would like to retrieve documents in the target set that are semantically associated with the source document. This can be very useful in the IT domain to suggest solutions from an archive for a new problem. Similarly in medical domain, it can also reveal typical treatments for some reported symptoms of a patient. While this problem can be solved by using a standard retrieval model by treating the source document as a long query to score and rank target documents, we will show that PTM can enhance such a method through bridging the vocabulary gap between the source document and a target document through the learned underlying topic mappings. Such an evaluation will help us indirectly measure the quality of the mining results of PTM. (Please be aware of that the goal of this experiment is not to find the best way for retrieving target documents.)

Specifically, if we use the source document (either ticket problem description or symptom de-

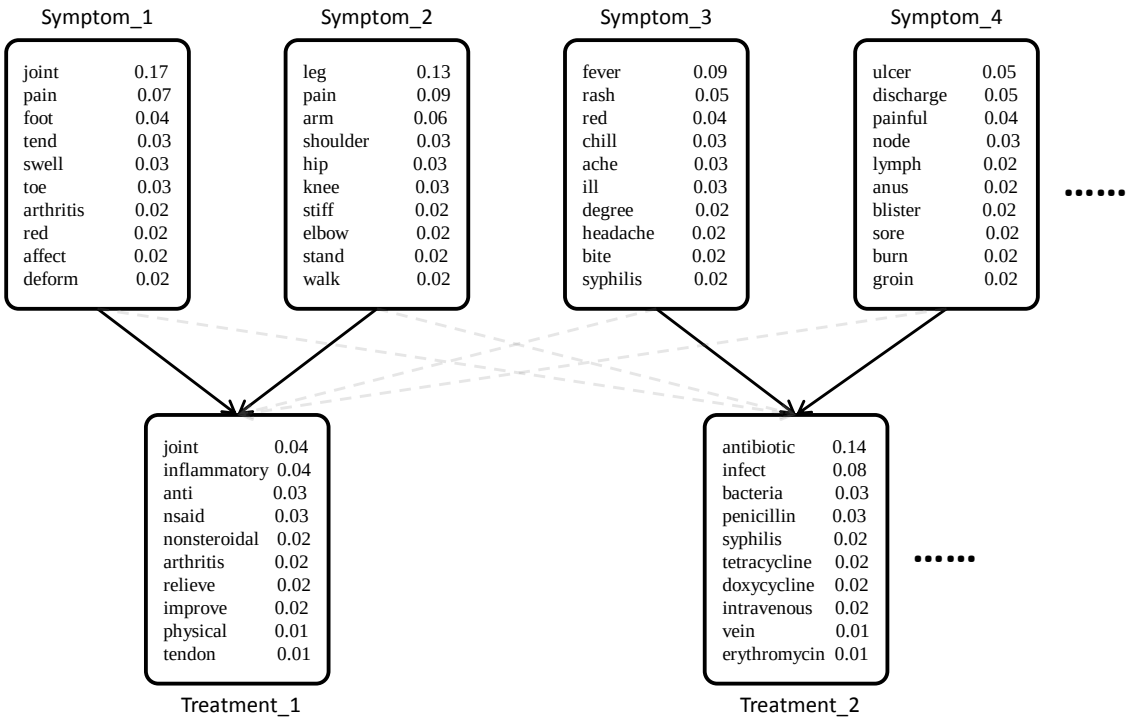


Figure 5.5: Word distributions and topic mapping learned from PTM on medical data ($K_s = 40, K_t = 20$)

scription) as a query q, retrieval methods like the KL-divergence retrieval model [31] can be used to rank possible targets (solutions or treatments). With this retrieval model, one critical issue is how to construct a query language model for q, and we will use PTM to improve the construction of the query language model. We will thus experiment with three different methods:

1. Baseline retrieval method: we define a query language model for q as $P(w|q) \propto c(w,q)$, where $c(w,q)$ is the number of words w in q.

2. PLSA based approach: one disadvantage of the baseline approach is that the source description may be too short (e.g. a short description of a patient's symptom) to be informative to retrieve the archived solutions. One possible solution is to use PLSA model to help expand the original query language model $P(w|q)$ [57]. Specifically, we first train PLSA model on a set of archived source documents and learn K_s from it. Then, we first use the folding-in method proposed in [27] to calculate $\{P(\theta_i|q)\}_{i=1}^{K_s}$ in the source description. Basically, we fix the parameters $\{P(w|\theta_i)\}_{i=1}^{K_s}$ which are estimated from the training data, and then run the EM algorithm used in PLSA to estimate $\{P(\theta_i|q)\}_{i=1}^{K_s}$. Then, the smoothed query language

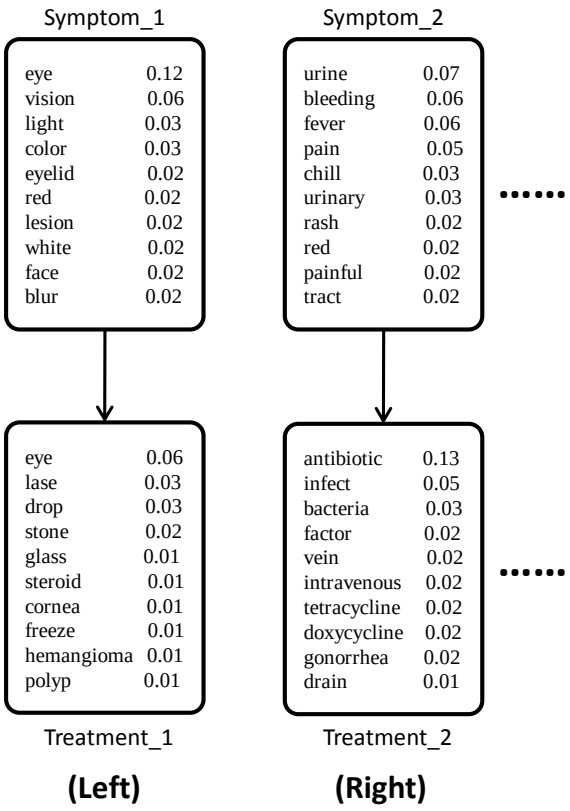


Figure 5.6: Word distributions and topic mapping learned from APTM on medical data where $K = 20$

model is calculated as follows:

$$P_{PLSA}(w|q) = \lambda P(w|q) + (1-\lambda)\sum_{i=1}^{K_s} P(w|\theta_i)P(\theta_i|q)$$

where $P(w|q)$ is the original query language model. It can be easily verified that:

$$\sum_{w} P_{PLSA}(w|q) = 1$$

3. PTM based approach: Both the baseline method and the PLSA based approach would fail if the query document (source document) does not have much overlap in vocabulary with a target document (which is often the case as, e.g., treatments and symptoms are often described in different terms). Our PTM model can be used to solve or alleviate the problem through expanding the query through related topics and their mappings to the target topics,

which presumably helps "crossing the source-target boundary". Specifically, first we train our PTM model on a training data set and learn all its parameters. Then, we still use the folding-in method in PLSA to estimate $\{P(\theta_i|q)\}_{i=1}^{K_s}$ of our source query q. After that, we use the topic mapping probabilities $\{P(\psi_j|\theta_i)\}_{i=1\ldots K_s}^{j=1\ldots K_t}$ to calculate the corresponding target topic mixture distribution for q, i.e. $\{P(\psi_j|q)\}_{j=1}^{K_t}$, using Eq. 5.2. Then, the smoothed query language model is calculated as follows:

$$P_{PTM}(w|q) \propto \lambda P(w|q) + (1-\lambda)\left(\sum_{i=1}^{K_s} P(w|\theta_i)P(\theta_i|q) + \sum_{j=1}^{K_t} P(w|\phi_j)P(\phi_j|q)\right),$$

Both the parameters $\{P(w|\theta_i)\}_{i=1}^{K_s}$ and $\{P(w|\phi_j)\}_{j=1}^{K_t}$ are got from our training data.

Since there is no relevance judgement data available for evaluating the retrieval performance, we conduct our experiments in a simulated way. Specifically, first we split our data into two parts, training and testing, and then learn either PLSA or PTM model on the training data. Second, we treat the source document in each testing parallel document pair as a query, and treat its corresponding target document as the true answer for the retrieval. After that, we put all the target documents into a pool, including target documents in both training and testing parallel document pairs, and using the three methods to rank all the targets based on each source query. We use the rank of the true targets to evaluate these three methods. For each method, the higher it ranks the true target, the better that method is. Since we use the KL-divergence retrieval model [31] for our purpose, we also build a document language model η_t for each target d_t in the archive $P(w|\eta_t) \propto c(w, d_t)$. Then, we score a target d_t based on the KL-divergence between the different query language models and η_t.

Two data sets are used for this evaluation. In the first data set, we randomly select 2500 tickets as our training data, and randomly select another 200 tickets as our test data. For each test ticket, we use its problem field as a source query to retrieve relevant tickets from those 2700 solution/target documents. In the second data set, we randomly split our 1300 medical document pairs into two parts: 1200 for training and 100 for testing. The number of source and target topics are set empirically.

To analyze the effectiveness of PTM in bridging the vocabulary gap between source and target

		IT		Medical	
		Easy	**Difficult**	**Easy**	**Difficult**
Baseline		23.23	639.47	4.14	216.15
PLSA	**0.8**	24.42	645.07	3.69	213.24
	0.85	23.19	652.12	3.66	213.87
	0.9	23.56	653.59	3.79	214.56
	0.91	22.91	653.07	3.83	215.1
	0.95	23.23	656.77	3.93	215.75
PTM	**0.8**	29.86	519.6	4.55	174.92
	0.85	27.96	529.79	4.45	181.41
	0.9	25.91	542.15	4.1	192.27
	0.91	25.51	547.43	4.07	195.1
	0.95	24.46	573.81	4.1	205.66

Figure 5.7: Effectiveness of PTM in improving difficult cases for document matching

document sets, we divide all the test cases into two groups: easy and difficult. If one test case's source document and target document do not have vocabulary gap and are very similar to each other, then the test case will be put into the easy group, otherwise it will be put into the hard group. Here, we use cosine similarity to compare the similarity between a source document and a target document. If the similarity is above 0.2, then we regard that there is not significant vocabulary gap between the two documents. By this way, there are 80 test cases in IT service data and 39 test cases in Medical data are classified into easy groups, and all the other test cases are classified into the hard groups. Then, for each group, we calculate the average rank of the true target of all the test cases in this group. Here, we use average rank as the metric, because, compared with the commonly used measure MRR (Mean Reciprocal Rank), average rank allows us to see more clearly the improvement on the difficult cases. Figure 5.7 shows the experimental results. For both PTM and PLSA, we also vary the value of the combination weight λ, which is list in the second column. Here, λ controls the weight of the original source document in the expanded query. If it is set too small, both PLSA and PTM will get worst results because the expanded query is too different from the original source document.

From the table, we can see that PTM indeed improves over the baseline for difficult cases consistently. However, its performance on the easy cases is often not as good as the baseline, which is expected because by bringing latent topics to smooth query language model, we may lose discriminativeness. Indeed, the trend along the parameter variation shows clearly this tradeoff: if we trust the baseline model more, we would do better on easy cases, but worse on difficult cases,

thus the potential for PTM to improve difficult cases is very clear. With appropriate setting of the weighting parameter, it's possible for PTM to do better for both easy cases and difficult cases. However, on the other hand, we did not see clear improvement of PLSA on the difficult cases over the baseline method, though there is a little improvement over the easy cases with some settings of the weighting parameter. This indicates that smoothing the query language model with only the source topics is not effective enough for bridging the vocabulary gaps between the source and target documents.

5.4.4 Efficiency Analysis

In this section, we evaluate the efficiency of our PTM model in two ways. First, we compare the performance of our model with a PLSA based method, in terms of CPU time and memory usage. This test is conducted on a single node, and it's mainly used to show whether using PTM to mine both topics and their mappings will cost more time or space, compared with a PLSA based method which only mines two sets of topics independently. Second, we exam the scalability of our model on millions of pair documents base on Hadoop framework, which is to test the feasibility of using our model for real business where a large collection of parallel documents needs to be analyzed. All the efficiency experiments are conducted on IT service ticket collections.

Experiments on a Single Node

Experimental Setup

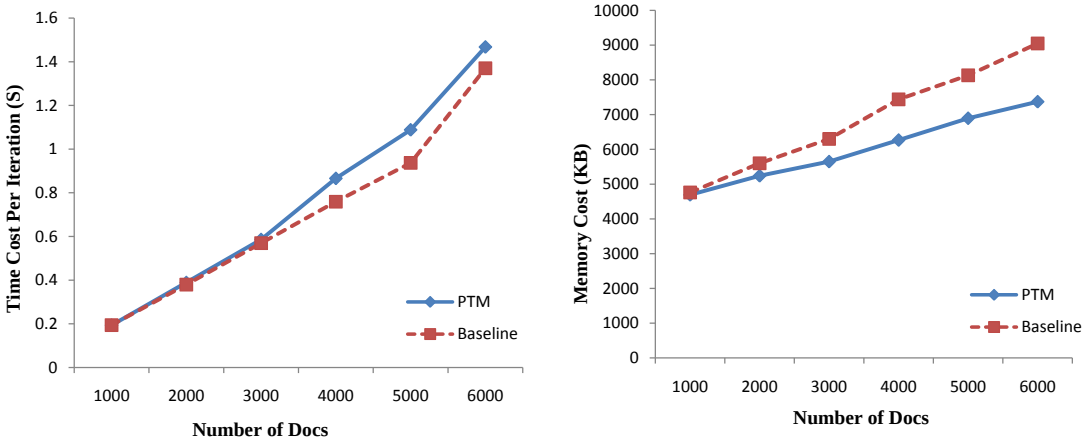
We use the following performance metrics for evaluation: (1) **CPU time**: the average time spent on one iteration of EM algorithm; (2) **memory usage**: the maximum memory usage during the execution. We use PLSA model as the baseline algorithm. For comparing CPU time, PLSA runs on the problem corpus and the solution corpus separately, and then uses the sum of average time spent per iteration on both corpus as the CPU time. Similarly, for comparing memory, we also sum the memory usage of PLSA on both corpus together as its memory usage.

There are two main factors in the PTM models which affect its total cost: the total number of document pairs M and the number of topics K_s and K_t. All the single node experiments are performed on a machine with 1GB RAM and 3.2 GHz CPU.

Scalable to the number of documents

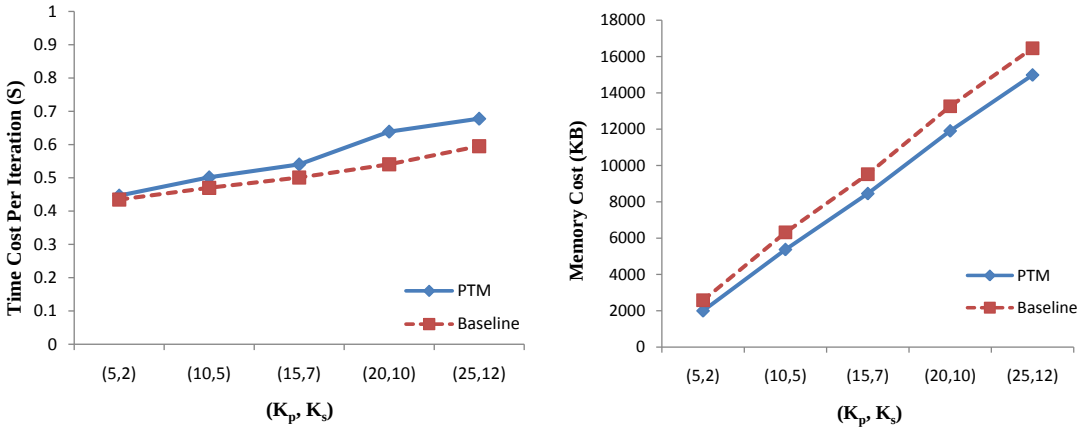
We randomly vary the corpus size from1000 to 6000 document pairs (tickets), while setting all the other parameters as the following: the number of problem and solution topics as 10 and 5, respectively, the number of unique words is 9469. For each trial, we run PTM and PLSA 10 times and compute the average time per iteration. For the memory usage, we use the maximum memory usage during the entire run. Since there is some common memory cost of both PTM and PLSA (e.g. the memory cost of loading the inverted index of the whole corpus), we deduct that part from the results and only compare the actual memory usage of each algorithm.

Figure 5.8 (a) plots CPU time as a function of the number of documents. We can see that the time spent for PTM per iteration is almost the same as baseline PLSA model, even though PTM mines more patterns such as topic mapping probabilities. From Figure 5.8 (b), we find that the maximum memory usage for the baseline method is a little larger than the PTM model. This is because in the baseline method, the PLSA model needs to store two different sets of topic portion parameters $\{P(\theta_i|s)\}$ and $\{P(\psi_j|t)\}$ parameters for both source and target collections. On the other hand, PTM only stores one set of topic portion parameters $\{P(\theta_i|s)\}$ and one set of topic correlation parameters $\{P(\psi_j|\theta_i)\}$ which do not depend on the number of documents. Since the number of topics is always much smaller than the number of documents, the total memory usage of PTM is smaller than the baseline method.



(a) CPU time (b) Memory Usage

Figure 5.8: Efficiency Analysis with Different Corpus Size



(a) CPU time (b) Memory Usage

Figure 5.9: Efficiency Analysis with Different Topics

Scalable to the number of topics

In this experiment, we test our algorithm's efficiency by changing the number of topics, while fixing the other parameters, e.g. the size of document corpus is 2500. We set the number of target topics K_t as the half of the number of source topics.

Figure 5.9 shows the experimental results, from which we get similar conclusion as from Figure 5.8. We also notice that both the CPU time and memory usage increase linearly as the number of topics. In summary, despite PTM provides richer mining results compared with traditional topic models, it does not sacrifice any performance in CPU time and memory usage.

Experiments on Multiple Nodes

Since the PTM model is estimated using EM algorithm, it is possible to scale up in parallel using Map-Reduce framework [15]. As a result, our algorithm can mine millions or even billions of parallel document collections. The documents are first distributed into different mappers, which calculate all the hidden variables (E-step: Eq. 5.6 to Eq. 5.8). Then, reducers will collect these calculation results from mappers and update all the model parameters (M-step). We implement the Map-Reduce version of our algorithm with Hadoop [2], and test it on a set of 3 million tickets. The experiments are conducted on a cloud computing testbed which consists of 128 HP DL160 compute nodes with dual quad core CPUs (2.66GHz), 16GB of RAM, and 2 TB of disk space.

[2] http://hadoop.apache.org

The performance metrics we use are 1) speed-up that is the ratio between running time on one node and that on N nodes, 2) execution time per EM iteration.

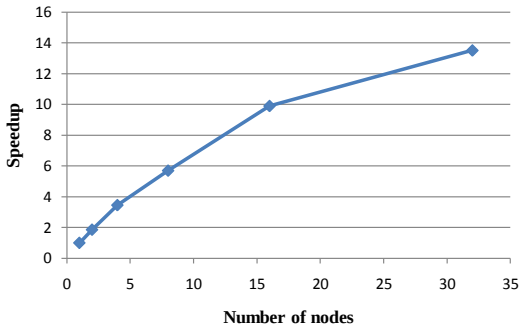
Figure 5.10(a) shows how well our algorithm is scaled up with different number of nodes. We observe a near linear scale-up initially, then goes flat, because the mapper jobs are getting too small and MapReduce overhead overtakes. However, as we have more data, the linear scale-up will extend to a larger number of nodes.

We also test the efficiency of estimating our model with different number of mappers and reducers. The performance is evaluated by the time cost for each iteration of the EM algorithm. Figure 5.10(b) shows the time cost of our algorithm with different number of mappers, where the number of reducers is set to 5, and Figure 5.10(c) shows the time cost of our algorithm with different number of reducers, where the number of mappers is fix to 200. We observe that with 200 mappers and 5 reducers, one iteration of our EM algorithm takes about 60 to 80 seconds when we train our model over 3 million tickets. Another finding is that increasing the number of reducers does not save the computation cost, while increasing the number of mappers does. It indicates that in the Map-Reduce version of PTM model, most of the computation cost of our algorithm is done during E-step rather than M-step.

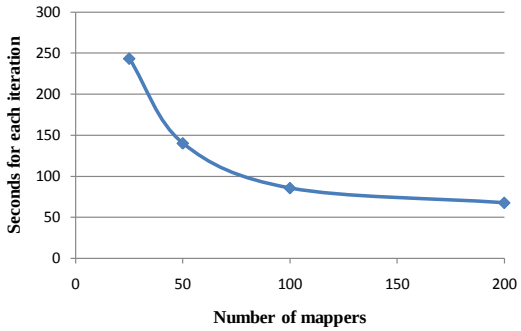
Theoretically, on a single node, the time complexity of each EM iteration for estimating PTM is $O((K_s + K_t)MN_{avg})$, where M is the totaly number of tickets and N_{avg} is the average number of unique words in each ticket. On X multiple nodes, the time complexity for each EM iteration will be approximately $O((K_s + K_t)\frac{M}{X}N_{avg})$ plus additional cost by Hadoop framework.

5.5 Conclusions and Future Work

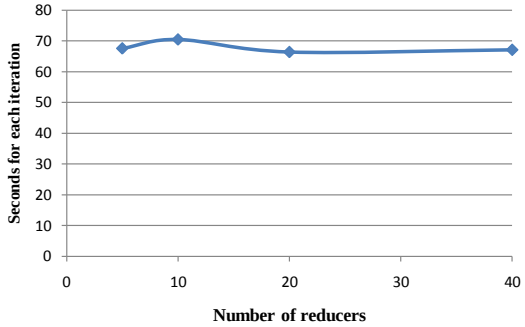
In this chapter, we proposed a novel approach called Probabilistic Topic Mapping (PTM) model for mining two text fields, *i.e.* a parallel document collection, of a MDT database. Our experimental results show that PTM can effectively discover meaningful topics and their mappings from parallel document collections. We also use applications to demonstrate PTM's capability of improving text matching and retrieval when there is vocabulary gap. The Map-Reduce version of our PTM makes it feasible to analyze parallel collections with million parallel documents. The proposed PTM model can be regarded as a novel extension of the PLSA model [26] to model parallel documents



(a) Speedup with different number of nodes



(b) Efficiency with different number of mappers



(c) Efficiency with different number of reducers

Figure 5.10: Scale up analysis of PTM model

and discover topic mappings. Similar extension can also be made to the LDA model [9], which would be an interesting future research direction. Also, in many applications, the MDT databases are increasing over time, so how to modify PTM to model the time evolved topic mapping between two text fields is also a promising direction.

Chapter 6

Conclusions

In this chapter, I will summarize the contributions of my thesis and discuss some future research problems that are interesting to explore.

6.1 Summary

As the amount of MultiDimensional Text databases grows explosively, efficient and effective analysis of this kind of databases and mining knowledge out of them becomes more and more desirable. In my thesis, I first identified three major challenging tasks in analyzing MDT databases, namely digestion, exploration, and analysis, and then proposed and studied different novel models to solve research problems raised by these challenging tasks. Specifically:

1. To support efficient digestion of the text information within an MDT database, I studied a novel data model called Topic Cube. It combines OLAP technology and topic modeling approach so that we can extend OLAP to the text dimension which allows an analyst to flexibly digest the content in text documents together with other standard dimensions. Technically, a topic cube extends the standard data cube in two ways: (1) adopt a hierarchical topic tree to define a topic dimension for exploring text information, and (2) store word distributions as the primary content measure (and topic coverage probabilities as auxiliary measure) of text information. To efficiently materialize topic cube, I proposed two kinds of heuristic aggregations which leverage previously estimated models in component cells or lower level topics to choose a good starting point for estimating the model for a merged large cell or higher level topics. Experimental results show that the heuristic aggregations are effective and topic cube can be used for many different applications.

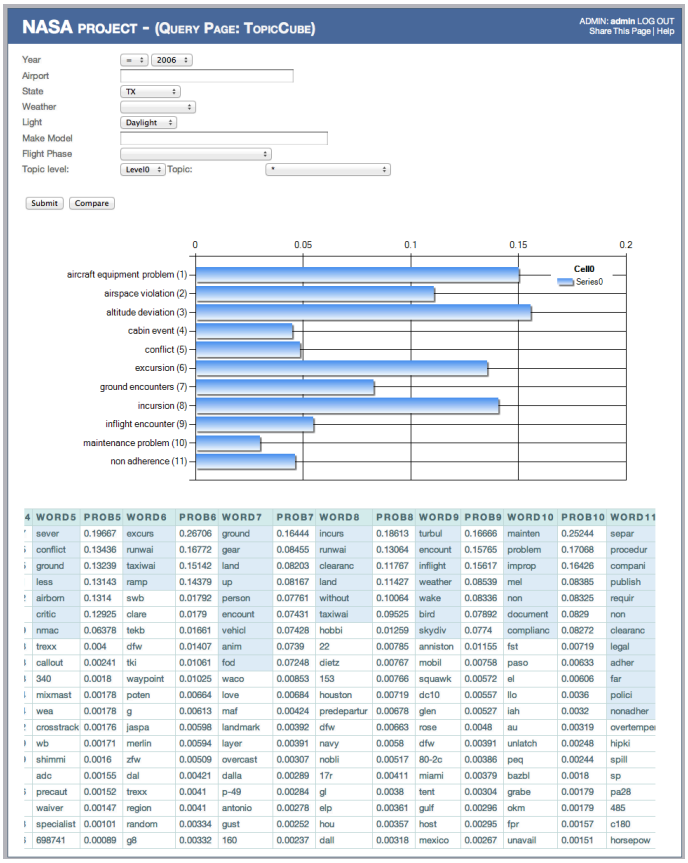
2. To support flexible and efficient online exploration of MDT databases, I proposed a novel cube called MicroTextCluster Cube (*MiTexCube*). I proposed a progressive materialization algorithm

for this novel cube, an update algorithm for the cube when there are changes in the database, and methods to leverage *MiTexCube* for three analysis tasks, including standard cell summarization, query-specific cell summarization, and common topic coverage comparison. Experimental results on real multidimensional text databases show that applications based on the proposed materialized *MiTexCube* are more efficient than the baseline methods of direct analysis based on document units in each cell, without sacrificing much quality of analysis. The proposed *MiTexCube* has several parameters to accommodate flexible tradeoffs between time and space as well as effectiveness and efficiency. We conducted several experiments to understand the effects of changing these parameters and discussed how to set these parameters empirically.

3. To carry out comparative analysis on MDT databases with multiple text fields, I proposed and studied a novel model called Probabilistic Topic Mapping (PTM) model. It simultaneously mines two different sets of topics from two parallel text fields and generates their probabilistic mappings as well. Experimental results show that PTM can effectively discover meaningful topics and their mappings from parallel document collections. We also use applications to demonstrate PTM's capability of improving text matching and retrieval when there is vocabulary gap. The Map-Reduce version of our PTM makes it feasible to analyze parallel collections with million parallel documents. Although the PTM model is originally designed for MDT databases with multiple text fields, an MDT database with only one text field can also be mined by the model if we first generate a parallel document collection based on the standard dimensions of the database.

To demonstrate the power of the proposed techniques, as part of my thesis work, both Topic Cube and *MiTexCube* have already been implemented in a domain independent prototype system called EventCube (http://dmserv1.cs.illinois.edu/eventcube/). Figure 6.1 shows an example of the Topic Cube function in the EventCube system, in which a user can specify values in standard dimensions to digest the text information online.

In summary, the main contribution of this thesis is to systematically advance the state-of-the-art technology for supporting digestion, exploration, and analysis in multidimensional text databases. All the studied algorithms are general and thus can be applied to any multidimensional text databases in any application domain.



WORD5	PROB5	WORD6	PROB6	WORD7	PROB7	WORD8	PROB8	WORD9	PROB9	WORD10	PROB10	WORD11
sever	0.19667	excurs	0.26706	ground	0.16444	incurs	0.18613	turbul	0.16666	mainten	0.25244	separ
conflict	0.13436	runwai	0.16772	gear	0.08455	runwai	0.13064	encount	0.15765	problem	0.17068	procedur
ground	0.13239	taxiwai	0.15142	land	0.08203	clearanc	0.11767	inflight	0.15617	improp	0.16426	compani
less	0.13143	ramp	0.14379	up	0.08167	land	0.11427	weather	0.08539	mel	0.08385	publish
airborn	0.1314	swb	0.01792	person	0.07761	without	0.10064	wake	0.08336	non	0.08325	requir
critic	0.12925	clare	0.0179	encount	0.07431	taxiwai	0.09525	bird	0.07892	document	0.0829	non
nmac	0.06378	tekb	0.01661	vehicl	0.07428	hobbi	0.01259	skydiv	0.0774	complianc	0.08272	clearanc
trexx	0.004	dfw	0.01407	anim	0.0739	22	0.00785	anniston	0.01155	fst	0.00719	legal
callout	0.00241	tki	0.01061	fod	0.07248	dietz	0.00767	mobil	0.00758	paso	0.00833	adher
340	0.0018	waypoint	0.01025	waco	0.00853	153	0.00766	squawk	0.00572	el	0.00606	far
mixmast	0.00178	poten	0.00664	love	0.00684	houston	0.00719	dc10	0.00557	llo	0.0036	polici
wea	0.00178	g	0.00613	maf	0.00424	predepartur	0.00678	glen	0.00527	iah	0.0032	nonadher
crosstrack	0.00176	jaspa	0.00598	landmark	0.00392	dfw	0.00663	rose	0.0048	au	0.00319	overtempe
wb	0.00171	merlin	0.00594	layer	0.00391	navy	0.0058	dfw	0.00391	unlatch	0.00248	hipki
shimmi	0.0016	zfw	0.00509	overcast	0.00307	nobli	0.00517	80-2c	0.00386	peq	0.00244	spill
adc	0.00155	dal	0.00421	dalla	0.00289	17r	0.00411	miami	0.00379	bazbl	0.0018	sp
precaut	0.00152	trexx	0.0041	p-49	0.00284	gl	0.0038	tent	0.00304	grabe	0.00179	pa28
waiver	0.00147	region	0.0041	antonio	0.00278	elp	0.00361	gulf	0.00296	okm	0.00179	485
specialist	0.00101	random	0.00334	gust	0.00252	hou	0.00357	host	0.00295	fpr	0.00157	c180
698741	0.00089	g8	0.00332	160	0.00237	dall	0.00318	mexico	0.00267	unavail	0.00151	horsepow

Figure 6.1: Topic Cube Function in EventCube System

6.2 Future Work

Integrative management and analysis of multidimensional text databases is a relatively new topic, and this thesis represents some initial steps toward solving this important problem. There are several interesting directions to further extend the work of this thesis.

Other Possible Combinations of OLAP and Text Mining Techniques

My thesis proposed a novel Topic Cube model to combine the OLAP technology with topic modeling approaches in text mining, and a novel *MiTexCube* model to combine the OLAP technology with clustering algorithms in text mining. Both of them show the possibility and the great potential of combining these two general areas. Indeed, there are many other powerful text mining techniques studied in literature which can be integrated with OLAP technology to provide other powerful tools for analyzing MDT databases. For example, combining OLAP with some deep text understanding

techniques in NLP would be a very promising direction to explore.

Domain Independent Systems Built on the Proposed Techniques

Since all the proposed algorithms are general, we can develop general toolkits or systems to support multiple applications across different domains based on the proposed techniques. The EventCube system can support the digestion and exploration functions for MDT databases in any domain. It would be interesting to further extend the current system or build a new general system which can use PTM model to support comparative analysis on any MDT databases. For example, given an MDT database with only one text field, the system can first do some correlation analysis and find out the most meaningful combinations of the standard dimensions. Then, it can use these combinations to partition all the records into groups, align documents from different groups, and create the most interesting parallel document collections from the database, so that PTM can be used to mine the most valuable knowledge. Furthermore, it would also be interesting to connect and combine all the proposed algorithms to provide users a more powerful tool to analyze MDT databases.

Other Challenges for Integrative Text Mining in MDT databases

For many MDT databases, besides the structured dimensions and the text dimension, they also have an associated numerical value in each record. For example, in an MDT database in business domain, it always has a numerical dimension called "Sale" which stores the total number of a product sold in a certain period. Mining the highly correlated terms with the numerical values from the text data and using them to predict unknown numerical values would be very desirable and challenging. For example, in an MDT database with both user reviews and sales, a sales manager would like to predict the sale of a product based on current user reviews and then make corresponding decisions. Another challenging but interesting direction is to use the standard dimensions to supervise topic discovery in text data. For example, the "Time" dimension could guide topic discovery in an MDT to form a trend of topic evolution, while two highly correlated standard dimensions also imply that the associated documents should have correlated topics. All of these explorations will generate more interesting studies on integrative text mining in MDT databases.

References

[1] Anomaly event schema. http://www.asias.faa.gov/pls/portal/stage.meta_show_column?v_table_id=165477.

[2] Aviation safety reporting system. http://asrs.arc.nasa.gov/.

[3] The dblp computer science bibliography. http://www.informatik.uni-trier.de/~ley/db/.

[4] Megaputer's polyanalyst. http://www.megaputer.com/.

[5] S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. In *VLDB*, pages 506–521, 1996.

[6] N. Bansal and N. Koudas. Blogscope: a system for online analysis of high volume text streams. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 1410–1413. VLDB Endowment, 2007.

[7] D. Blei and J. Lafferty. Correlated topic models. In *NIPS '05: Advances in Neural Information Processing Systems 18*, 2005.

[8] D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *NIPS*, 2003.

[9] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[10] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98*, pages 335–336, New York, NY, USA, 1998. ACM.

[11] V. T. Chakaravarthy, H. Gupta, P. Roy, and M. Mohania. Efficiently linking text documents with relevant structured information. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 667–678. VLDB Endowment, 2006.

[12] S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *SIGMOD Rec.*, 26(1):65–74, 1997.

[13] B.-C. Chen, L. Chen, Y. Lin, and R. Ramakrishnan. Prediction cubes. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 982–993. VLDB Endowment, 2005.

[14] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams. In *Proc. 2002 Int. Conf. Very Large Data Bases (VLDB'02)*, pages 323–334, Hong Kong, China, Aug. 2002.

[15] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *NIPS*, pages 281–288, 2006.

[16] W. F. Cody, J. T. Kreulen, V. Krishna, and W. S. Spangler. The integration of business intelligence and knowledge management. *IBM Syst. J.*, 41(4):697–713, 2002.

[17] G. Cong, L. Wang, C.-Y. Lin, Y.-I. Song, and Y. Sun. Finding question-answer pairs from online forums. In *SIGIR '08*, pages 467–474, New York, NY, USA, 2008. ACM.

[18] A. Corrada-Emmanuel and W. B. Croft. Answer models for question answering passage retrieval. In *SIGIR '04*, pages 516–517, New York, NY, USA, 2004. ACM.

[19] H. T. Dang, D. Kelly, and J. Lin. Overview of the trec 2007 question answering track. In *Proceeding of the 16th Text Retrieval Conference*, 2007.

[20] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1:224–227, Feb. 1979.

[21] 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.

[22] F. M. fei Jiang, J. Pei, and A. W. chee Fu. Ix-cubes: iceberg cubes for data warehousing and olap on xml data. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 905–908, New York, NY, USA, 2007. ACM.

[23] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *ICDE*, 00:152, 1996.

[24] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[25] T. Hofmann. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In T. Dean, editor, *IJCAI*, pages 682–687. Morgan Kaufmann, 1999.

[26] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM Press, 1999.

[27] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.*, 42(1-2):177–196, 2001.

[28] A. Inokuchi and K. Takeda. A method for online analytical processing of text data. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 455–464, New York, NY, USA, 2007. ACM.

[29] P. G. Ipeirotis, A. Ntoulas, J. Cho, and L. Gravano. Modeling and managing changes in text databases. *ACM Trans. Database Syst.*, 32(3):14, 2007.

[30] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[31] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01*, pages 111–119, New York, NY, USA, 2001. ACM.

[32] C. X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao. Text cube: Computing ir measures for multidimensional text database analysis. In *ICDM*, pages 905–910, 2008.

[33] E. Lo, B. Kao, W.-S. Ho, S. D. Lee, C. K. Chui, and D. W. Cheung. Olap on sequence data. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 649–660, New York, NY, USA, 2008. ACM.

[34] Q. Mei, D. Cai, D. Zhang, and C. Zhai. Topic modeling with network regularization. In *WWW '08*, pages 101–110, New York, NY, USA, 2008. ACM.

[35] 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.

[36] Q. Mei, C. Liu, H. Su, and C. Zhai. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors, *WWW*, pages 533–542. ACM, 2006.

[37] Q. Mei, X. Shen, and C. Zhai. Automatic labeling of multinomial topic models. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 490.

[38] Q. Mei and C. Zhai. Generating impact-based summaries for scientific literature. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, page 816.

[39] D. Mimno, H. M. Wallach, J. Naradowsky, D. A. Smith, and A. Mccallum. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 880–889, Singapore, August 2009. Association for Computational Linguistics.

[40] R. M. Nallapati, A. Ahmed, E. P. Xing, and W. W. Cohen. Joint latent topic models for text and citations. In *KDD '08*, pages 542–550, New York, NY, USA, 2008. ACM.

[41] J. M. Pérez, R. Berlanga, M. J. Aramburu, and T. B. Pedersen. A relevance-extended multi-dimensional model for a data warehouse contextualized with documents. In *DOLAP '05: Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 19–28, New York, NY, USA, 2005. ACM.

[42] J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the ACM SIGIR'98*, pages 275–281, 1998.

[43] D. Ramage, P. Heymann, C. D. Manning, and H. Garcia-Molina. Clustering the tagged web. In *WSDM '09*, pages 54–63, New York, NY, USA, 2009. ACM.

[44] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[45] L. Shrestha and K. McKeown. Detection of question-answer pairs in email conversations. In *COLING '04*, page 889, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[46] A. Simitsis, A. Baid, Y. Sismanis, and B. Reinwald. Multidimensional content exploration. *Proc. VLDB Endow.*, 1(1):660–671, 2008.

[47] R. Soricut and E. Brill. Automatic question answering using the web: Beyond the factoid. *Inf. Retr.*, 9(2):191–206, 2006.

[48] 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.

[49] 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, New York, NY, USA, 2006. ACM.

[50] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 567–580, New York, NY, USA, 2008. ACM.

[51] I. Titov and R. McDonald. Modeling online reviews with multi-grain topic models. In *WWW '08*, pages 111–120, New York, NY, USA, 2008. ACM.

[52] X. Wang, C. Zhai, X. Hu, and R. Sproat. Mining correlated bursty topic patterns from coordinated text streams. In P. Berkhin, R. Caruana, and X. Wu, editors, *KDD*, pages 784–793. ACM, 2007.

[53] X. Wei and W. B. Croft. Lda-based document models for ad-hoc retrieval. In *SIGIR '06*, pages 178–185, New York, NY, USA, 2006. ACM.

[54] P. Wu, Y. Sismanis, and B. Reinwald. Towards keyword-driven analytical processing. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 617–628, New York, NY, USA, 2007. ACM.

[55] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *SIGIR '08*, pages 475–482, New York, NY, USA, 2008. ACM.

[56] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[57] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Tenth International Conference on Information and Knowledge Management (CIKM 2001)*, pages 403–410, 2001.

[58] D. Zhang, J. Sun, C. Zhai, A. Bose, and N. Anerousis. PTM: probabilistic topic mapping model for mining parallel document collections. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1653–1656, New York, NY, USA, 2010. ACM.

[59] D. Zhang, C. Zhai, and J. Han. Topic cube: Topic modeling for olap on multidimensional text databases. In *SDM'09: Proceedings of the Ninth SIAM International Conference on Data Mining*, pages 1124–1135. SIAM, 2009.

[60] D. Zhang, C. Zhai, and J. Han. Mitexcube: Microtextcluster cube for online analysis of text cells. In *CIDU*, pages 204–218, 2011.

[61] D. Zhang, C. Zhai, and J. Han. Mitexcube: Microtextcluster cube for online analysis of text cells and its applications. *Statistical Analysis and Data Mining*, 2012. DOI: 10.1002/sam.11159.

[62] D. Zhang, C. Zhai, J. Han, A. Srivastava, and N. Oza. Topic modeling for olap on multidimensional text databases: topic cube and its applications. *Stat. Anal. Data Min.*, 2:378–395, December 2009.

[63] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, 1996.

[64] D. Zhou, J. Bian, S. Zheng, H. Zha, and C. L. Giles. Exploring social annotations for information retrieval. In *WWW '08*, pages 715–724, New York, NY, USA, 2008. ACM.