Topic Modeling for OLAP on Multidimensional Text Databases: Topic Cube and its Applications

Duo Zhang[†] ChengXiang Zhai[†] Jiawei Han[†] Ashok Srivastava[‡] Nikunj Oza[‡]
{dzhang22, czhai, hanj}@cs.uiuc.edu, {ashok.n.srivastava, nikunj.c.oza}@nasa.gov

[†]Department of Computer Science, University of Illinois at Urbana Champaign

[‡]Intelligent Systems Division of the NASA Ames Research Center

Abstract

As the amount of textual information grows explosively in various kinds of business systems, it becomes more and more desirable to analyze both structured data records and unstructured text data simultaneously. While online analytical processing (OLAP) techniques have been proven very useful for analyzing and mining structured data, they face challenges in handling text data. On the other hand, probabilistic topic models are among the most effective approaches to latent topic analysis and mining on text data. In this paper, we propose a new data model called *topic cube* to combine OLAP with probabilistic topic modeling and enable OLAP on the dimension of text data in a multidimensional text database. Topic cube extends the traditional data cube to cope with a topic hierarchy and store probabilistic content measures of text documents learned through a probabilistic topic model. To materialize topic cubes efficiently, we propose two heuristic aggregations to speed up the iterative EM algorithm for estimating topic models by leveraging the models learned on component data cells to choose a good starting point for iteration. Experiment results show that these heuristic aggregations are much faster than the baseline method of computing each topic cube from scratch. We also discuss potential uses of topic cube and show sample experimental results.

1 Introduction

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 [4, 10, 16], which enables flexible interactive analysis of multidimensional data in different granularities. It has been widely applied to many different domains [15, 23, 33]. OLAP on data warehouses is mainly supported through data cubes [11, 12].

As unstructured text data grows quickly, 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. These text data often exist either in the character fields of data records or in a separate place with links to the data records through joinable common attributes. Thus conceptually we have both structured data and unstructured text data in a database. For convenience, we will refer to such a database as a *multidimensional text database*, to distinguish it from both the traditional relational databases and the text databases which consist primarily of text documents.

As argued convincingly in [13], 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. A text narrative usually contains hundreds of words.

Table 1: An example of text database in ASRS

ACN	Time	Airport	···	Light	Narrative
101285	199901	MSP	···	Daylight	Document 1
101286	199901	CKB	···	Night	Document 2
101291	199902	LAX	···	Dawn	Document 3

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 1.

In the tree shown in Figure 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."

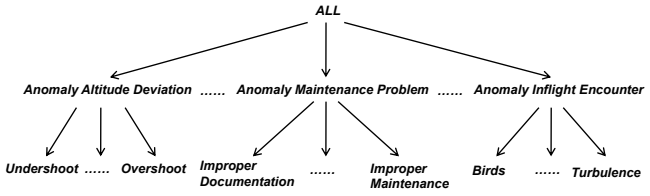


Figure 1: Hierarchical Topic Tree for Anomaly Event

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 [9, 20] and integrating text analysis with OLAP [19, 28], 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 [13], 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 effi-

ciently handle text dimension has not be addressed. We will review the related work in more detail in Section 2.

In this paper, 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 basic idea is to leverage probabilistic topic modeling [18, 8], 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 including hierarchical topic modeling [17, 7], author-topic analysis [31], spatiotemporal text mining [25], sentiment analysis [24], and multi-stream bursty pattern finding [34]. 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 [18] so that a data cube would carry parameters of a probabilistic model that can indicate the text content of the cell. Our basic 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 [29, 37], and can be further exploited to generate more informative summaries if needed. For example, in [27], 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 [26] or comparative summaries for comparing content across different contexts [24]. 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.

2 Related Work

To the best of our knowledge, no previous work has unified topic modeling with OLAP. However, some previous studies 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 [5], 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.

Text as character field: A major representative work in this

group is [35], 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 works to ours are BIW [13] and Polyanalyst [3]. 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 [13], only some high-level function descriptions are given with no algorithms given to efficiently support such OLAP operations on text dimension. Moreover in both works, 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). The topic model provides a principled way to map arbitrary text documents into topics specified in any topic hierarchy determined by an application *without* needing any training data. Previous work would mostly either need documents with known labels (for learning) which do not always exist, or cluster text documents in an unsupervised way, which does not necessarily produce meaningful clusters to an analyst, reducing the usefulness for performing OLAP on multidimensional text databases. We also propose heuristic methods for materializing the new topic cubes efficiently.

Text as component of OLAP Using OLAP technology to explore unstructured text data in a multidimensional text database has become a popular topic these years. In [30], 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 [22], a model called *text cube* is proposed, in which IR measures of terms are used to summarize the text data in a cell. In [38], we proposed a new model called *Topic Cube*. The difference between a topic cube and previous work is in that a topic cube has a topic dimension which allows users to effectively explore the unstructured text data from different semantic topics, and compared with frequent phrases or terms, such semantic topics are more meaningful to users. Furthermore, roll-up or drill-down along the topic dimension will also show users different granularities of topics. In this paper, we extend [38] by discussing and analyzing some new materialization strategies as well as new applications of a topic cube.

Topic models have been extensively studied in recent years [6, 7, 8, 17, 18, 24, 25, 26, 31, 34], all showing that they are very useful for analyzing latent topics in text and discovering topical patterns. However, all the work in this line only deals with pure text data. Our work can be regarded as a novel application of such models to support OLAP on multidimensional text databases.

3 Topic Cube as an Extension of Data Cube

The basic 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.

3.1 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 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. 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. 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) [18]. 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.

3.3 Definition of Topic Cube

DEFINITION 3.1. *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. 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 $p(topic|d_j)$.

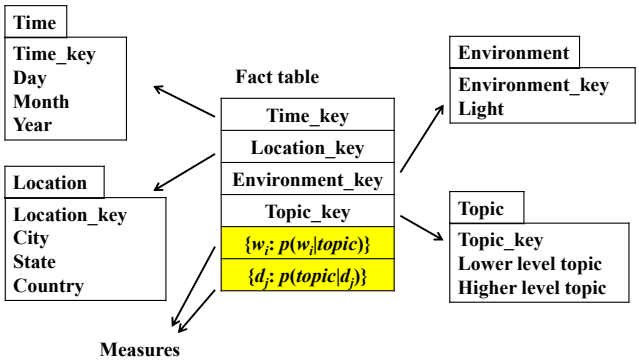


Figure 2: Star Schema of a Topic cube

Fig. 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. 1. For example, the left cuboid in Fig. 3 shows us word distributions of some finer

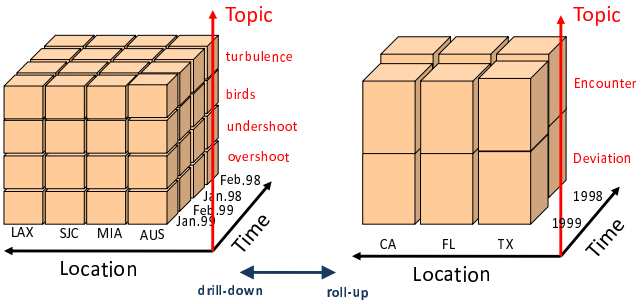


Figure 3: An example of a Topic Cube

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. 4, it shows two example cells of a topic cube (with only word distribution 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 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, 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, 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.

4 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, 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 where the "Location" field is "LAX" and the "Time" field is "Jan. 99". Then, for the second level of the hierarchical topic tree in Fig. 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.

4.1 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) [18] and latent Dirichlet allocation (LDA) [8]. 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.

4.1.1 Basic PLSA The basic PLSA [18] 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:

$$L(C|\Lambda) \propto \sum_{d \in C} \sum_{w \in V} c(w,d) \log \sum_{j=1}^{k} p(\theta_j|d) p(w|\theta_j) \tag{4.1}$$

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 [14]. 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:

$$p(z_{d,w} = j) = \frac{p^{(n)}(\theta_j|d) p^{(n)}(w|\theta_j)}{\sum_{j'=1}^{k} p^{(n)}(\theta_{j'}|d) p^{(n)}(w|\theta_{j'})} \tag{4.2}$$

M-step:

$$p^{(n+1)}(\theta_j|d) = \frac{\sum_w c(w,d) p(z_{d,w} = j)}{\sum_{j'} \sum_w c(w,d) p(z_{d,w} = j')} \tag{4.3}$$

$$p^{(n+1)}(w|\theta_j) = \frac{\sum_d c(w,d) p(z_{d,w} = j)}{\sum_{w'} \sum_d c(w',d) p(z_{d,w'} = j)} \tag{4.4}$$

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 [14].

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.

4.1.2 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 [32]. 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 "steers" the distribution to attract other words co-occurring with them, achieving the goal of extracting the content about this topic from text.

4.2 Materialization As discussed earlier, we use the PLSA model as our topic modeling method. To fully materialize a topic cube, we need to estimate PLSA and mine topics for each cell in the original data cube. 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 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 may need 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 maximum, it needs to start from a number of different random points to select the best one.

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

In general, measures in a data cube can be classified into three categories based on the difficulty in aggregation: distributive, algebraic, and holistic [12]. The measure in a topic cube is the word distributions of topics obtained from PLSA, which is a holistic measure. Therefore, there is no simple solution for us to aggregate measures from sub cells to compute the measures for super cells in a topic cube.

To solve this challenge, we propose to use a heuristic method to materialize a topic cube more efficiently in a bottom-up manner. The key idea is to leverage the topics discovered for sub cells to obtain a good starting point for discovering topics in a super cell with PLSA. Since estimation of PLSA is based on an iterative algorithm, an improved starting point leads to quicker convergence, thus speeding up the materialization of topic cube.

The same idea can also be used in a top-down manner, but it is generally less effective because it is much more difficult to infer specialized topics from general topics than to do the opposite since an interpolation of specialized topics would give us a reasonable approximation of a general topic. However, the top-down strategy may have other advantages over the bottom-up strategy, which we will further discuss in the next section after presenting the bottom-up strategy in detail.

5 Algorithms for Materialization of Topic Cube

The bottom-up strategy constructs a topic cube by first computing topics in small sub cells and then aggregating them to compute topics in larger super cells. Such a heuristic materialization algorithm can be implemented with two complementary ways to aggregate topics: *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, but they can also be applied together to further optimize the efficiency of materialization.

In both cases of aggregation, the basic idea is to iteratively materialize each cell starting with the bottom-level sub-cells, and When we mine topics from the documents of any cell in *GDC*, instead of running the EM algorithm randomly with multiple starting points and trying to find the best estimate, we would first aggregate the word distributions of topics in its sub cells to generate a good starting point, and then let the EM algorithm start from this starting point. Assuming that the estimates of topics in the sub-cells are close to the global maxima, we can expect this starting point to be close to the global maximum for the current cell. Thus, the EM algorithm can be expected to converge very quickly, and we do not need to restart the EM algorithm from multiple different random starting points. The two ways of aggregation mainly differ in the order to enumerate all the cells and will be further discussed in detail in the following subsections.

5.1 Aggregation in standard dimensions

When we materialize a topic cube through aggregation in standard dimensions, we would start with bottom cells and graduatelly reach larger cells by aggregating along each of the standard dimensions. The outline of this algorithm is shown in Table 2.

Table 2: 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.

Formally, 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, suppose we have already obtained word distributions $\{p_{c_i}(w|\theta_1^{(L_i)}), \ldots, p_{c_i}(w|\theta_{k_i}^{(L_i)})\}$ for each sub cell c_i, and we now want to mine topics $\{\theta_1^{(L_i)}, \theta_2^{(L_i)}, \ldots, \theta_{k_i}^{(L_i)}\}$ from S_{c_a}. The basic idea of our approach is when we apply the EM algorithm to mine these topics from S_{c_a}, we would start from a *good starting point*, which is obtained from aggregating word distributions for 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:

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

$$p_{c_a}^{(0)}(\theta_j^{(L_i)}|d) = p_{c_i}(\theta_j^{(L_i)}|d), \qquad if\ d \in S_{c_i} \tag{5.5}$$

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.

5.2 Aggregation in topic dimension Similarly, the basic 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 in a bottom-up manner, we would compute 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 again to find a good starting point to run the EM algorithm so that it can converge more quickly to an optimal esitmate. The outline of this algorithm is shown in Table 3.

Basically, 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.

Table 3: 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}

Formally, suppose cell c has a set of documents S associated with it, and we have obtained 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 $L+1$, given by:

$$Children(\theta_i^{(L)}) = \{\theta_{s_i}^{(L+1)}, \theta_{s_i+1}^{(L+1)}, \ldots, \theta_{e_i}^{(L+1)}\}, \forall 1 \le i \le k_L,$$

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

We can use the following formulas to compute a good starting point to run the EM algorithm for estimating parameters:

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

$$p_c^{(0)}(\theta_i^{(L)}|d) = \sum_{j' \in \{\theta_{s_i}^{(L+1)}, \ldots, \theta_{e_i}^{(L+1)}\}} p(\theta_{j'}|d), \qquad \forall d \in S \tag{5.6}$$

The intuition is similar: we pool together the expected counts of a word from all the sub topics of a super topic to get an overall count of the word for estimating the word distribution of the super topic. After the initial distribution is calculated, we can run the EM algorithm, which would converge much more quickly than starting from a random initialization since the initialization point obtained through aggregation is presumably already very close to the optimal estimate.

5.3 Combination of two aggregation strategies The two aggregation strategies described above can be used independently to speed up the materialization of a topic cube. Interestingly, they can also be combined together to further improve the efficiency of materializing a topic cube. Depending on 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 would first exhaustively mine 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 materialize all the other cells through aggregation along the standard dimensions to construct a topic cube. This combination strategy would lead to an algorithm very similar to the one shown in Table 2 except that we would change Step 2 to "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 would first exhaustively mine 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 topics of other levels in the hierarchial topic tree. The algorithm for this combination can be illustrated by modifying Table 3 so that we would change Step 2 to "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 can be done in parallel.

5.4 Saving Storage Cost One critical issue about topic cube is its storage. As discussed in Section 4.2, 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 store only 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.

5.5 Partial Materialization in Top-Down Strategy Another possible strategy to materialize a topic cube is to compute it in a top-down manner. Specifically, with this strategy we would first mine all the topics in the hierarchical tree from the largest cell (also called apex cell) in *GDC*. Then, in order to compute the topics in the sub cells of an apex cell, we would use 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 are materialized, we can further compute the sub sub cells of the apex

cell, and the word distributions of topics in their respective 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 in a top-down strategy, we would 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.

Since the materialization of a topic cube starts from the largest cell rather than the smallest cells in *GDC*, one advantage of the top-down strategy over the bottom-up strategy is that 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, thus no need to materialize. 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 the value for a small number of a_i's and set all the other dimensions as "*". Indeed, it is generally 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.

However, the idea of leveraging the word distributions obtained for existing cells to obtain a good starting point for the EM algorithm would not work as well as in the case of bottom-up materialization, which is a main disadvantage of the top-down strategy. Indeed, using the word distributions of topics in a super cell as the starting points for mining topics in its sub cells is generally not ineffective because a super cell is generally made of a number of sub cells, and the topics embedded in the documents of a single sub cell can 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 would likely be dominated by the other sub cells of the super cell. As a result, the word distributions of these topics would not be much different from random starting points for sub cell A. In contrast, in the bottom-up strategy, the starting point (e.g. calculated by Eq. 5.5) during estimating the topics in a super cell is a weighted combination of topics in all the sub cells, which can be expected to approximate an optimal estimate well.

6 Experiments

In this section, we present our evaluation of the topic cube model. First, we compare the computation efficiency of the proposed heuristic materialization methods with a baseline method which materializes a topic cube exhaustively cell by cell. Next, we will show several usages of topic cube to demonstrate its great potential of applications.

6.1 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*}. A summary of the number of documents in each base cell is shown in Table 4.

Table 4: 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 are used in our experiment: 6 topics in the first level and 16 topics in the second level as shown in Fig. 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 5: Hierarchical Topic Tree used in the Experiments

6.2 Efficiency Comparison In this section, we evaluate the efficiency of the two heuristic aggregation strategies proposed in Section 5 and compare each with the corresponding baseline method. For each aggregation method, we compare three strategies of constructing a topic cube.
(1) The heuristic aggregation method we proposed (either aggregation along the topic dimension or aggregation along the standard dimensions), which will be represented as *Agg*.
(2) An approximation method which only stores top k words in the word distribution of each topic and will be represented as *App*. The purpose of this method is to test the storage-saving strategy proposed in Section 5.4. For example, in 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, but with only top k words, in its sub cells to get a good starting point. Then, we initialize the EM algorithm with this starting point and run it until convergence. Similarly, in 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 experiments, we set the constant k to 10.
(3) The baseline method, which initializes the EM algorithm with random points and will be represented as *Rdm*. As stated before, the exhaustive method to materialize a topic cube runs EM algorithm by starting from several different random points and then selecting 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 is not surprising that the heuristic aggregation method is faster than the exhaustive method. Thus in our experiment, we use both the *average performance* and *best performance from a single 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 performance is computed using the *single* best run that converges to the best local optimum point (highest log likelihood) among these M runs.

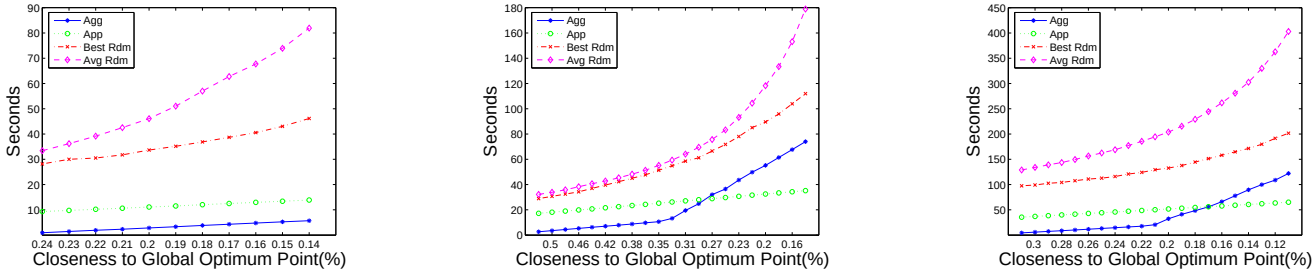
To measure the efficiency of these strategies, we look at how long it takes for these strategies to get to the same closeness to the global optimum point. Here, we assume that the convergence point of the best run of the M runs is the global optimum point. The experimental results are shown in Fig. 6. 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 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 experiments 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 horizontal 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 stable than *Agg*. For example, in Fig. 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 5 shows the log likelihood of the starting points of the three strategies. Here, the log likelihood of the objective function is calculated by Eq. (4.1). This value indicates how likely the documents are generated by topic models, so 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, further supporting our hypothesis that the starting points used in the proposed heuristic methods are closer to the optimum point than a random starting point, and thus our mehtods need less time to converge.

Table 5: 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

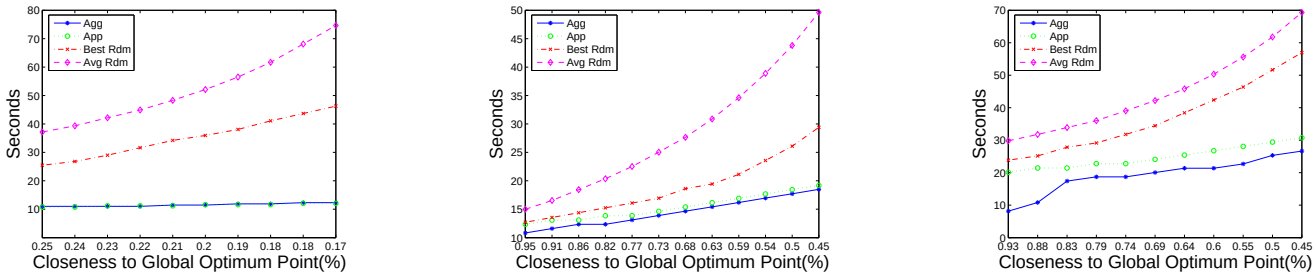


(a) Cell=(1999, CA, *) with 629 documents

(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 6: Efficiency comparison of different Strategies for aggregation in standard dimensions (top) and topic dimension (bottom).

6.3 Topic Comparison in Different Context One major application of a topic cube is to allow users to explore and analyze topics in different contexts. Here, we regard all the standard dimensions as contexts for topics. Fig. 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 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 7: Application of Topic Cube in ASRS

6.4 Topic Coverage in Different Context Topic coverage analysis is another application 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. These parameters indicate the topic coverage in 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. 8 shows the topic coverage analysis on our experiment data set. Fig. 8(a) is the topic coverage over different places and Fig. 8(b) is the topic coverage over different environment. With this kind of analysis, we can easily find out answers to questions such as "what is the most severe

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. 8 reveals some very interesting facts. Flights in Texas have more "turbulence" problems than in California and Florida, while Florida has the most severe "Encounter: Airborne" problem among these three places, and there is no evident difference of the coverage of anomalies such as "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.

6.5 Shaping Factor Analysis Analyzing the shaping factors of human performance during flights plays an important role in aviation safety research. The reporters of anomalous events tend to describe their physical factors, attitude, pressure, proficiency, preoccupation, etc., in the text reports, which can be potential causes of an anomalous event. Thus, it's necessary to analyze all these factors in text and their correlations with anomalies. 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 6 shows five different shaping factors as well as some examples and keywords describing them.

To support shaping factor analysis, we first extract keyword lists for describing shaping factors as follows. First, 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, information gain [36] is used to compute a score for each unigram and 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 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 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 consistent with our common sense. On the other hand, if we look into the difference of the correlations among different contexts, we can also make some interesting observations. 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 need to improve their communication environment.

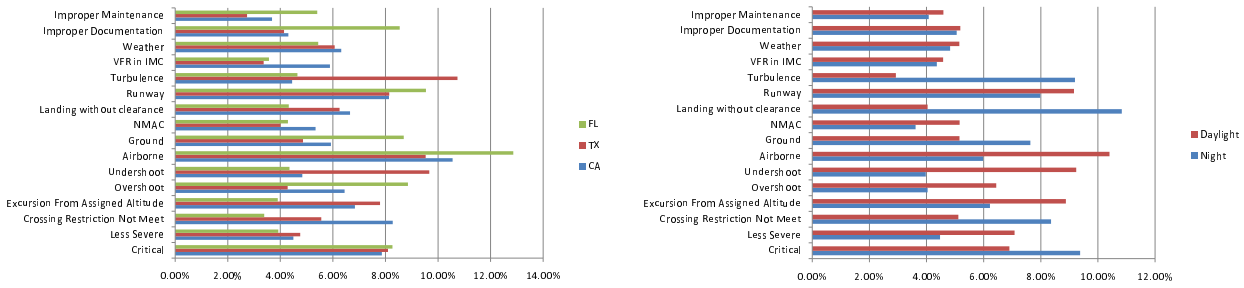
6.6 Accuracy of Categorization In this experiment, we test how accurate the topic modeling method is for document categorization. Since we only have our prior for each topic without training examples in our data set, we do not compare our method with supervised classification. Instead, we use the following method as our baseline. First, we use the prior of each topic to estimate a language model ϕ_j for each topic j. Then, we create a document language model ζ_d for each document with 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 $p(w|C) = \frac{c(w,C)}{\sum_{w'} c(w',C)}$ is the collection background model. Finally, we can use the negative KL-divergence [21] function to measure the similarity between a document d and a topic j: $S = -D(\phi_j||\zeta_d) = \sum_w p(w|\phi_j) \log \frac{p(w|\zeta_d)}{p(w|\phi_j)}$. If document d has a similarity score S higher than a threshold δ with a topic j, it would be classified into topic j. 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. The idea is to see whether the estimated topic model (i.e., θ_j) in a topic cube works better for categorizing documents than the original topic language model ϕ_j.

Our experiment is conducted on the whole data set, and we used the first level of topics in Fig. 5 as the target categories, i.e. we classify the documents into 6 categories. The gold standard is the "Anomaly Event" labels provided in the ASRS data. We plot the recall-precision curves by changing the value of the threshold δ in Fig. 10. From this figure, we can see that the curve of PLSA is above the baseline method, meaning that the topic representation obtained in the topic cube through PLSA can indeed achieve better categorization results than if we only use the prior knowledge about topics.

7 Conclusions

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 paper, we proposed a new data model (i.e.,

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



(a) Place Comparison

(b) Environment Comparison

Figure 8: Topic Coverage Comparison among Different Contexts

Table 6: 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

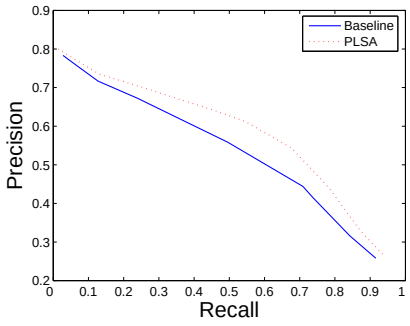
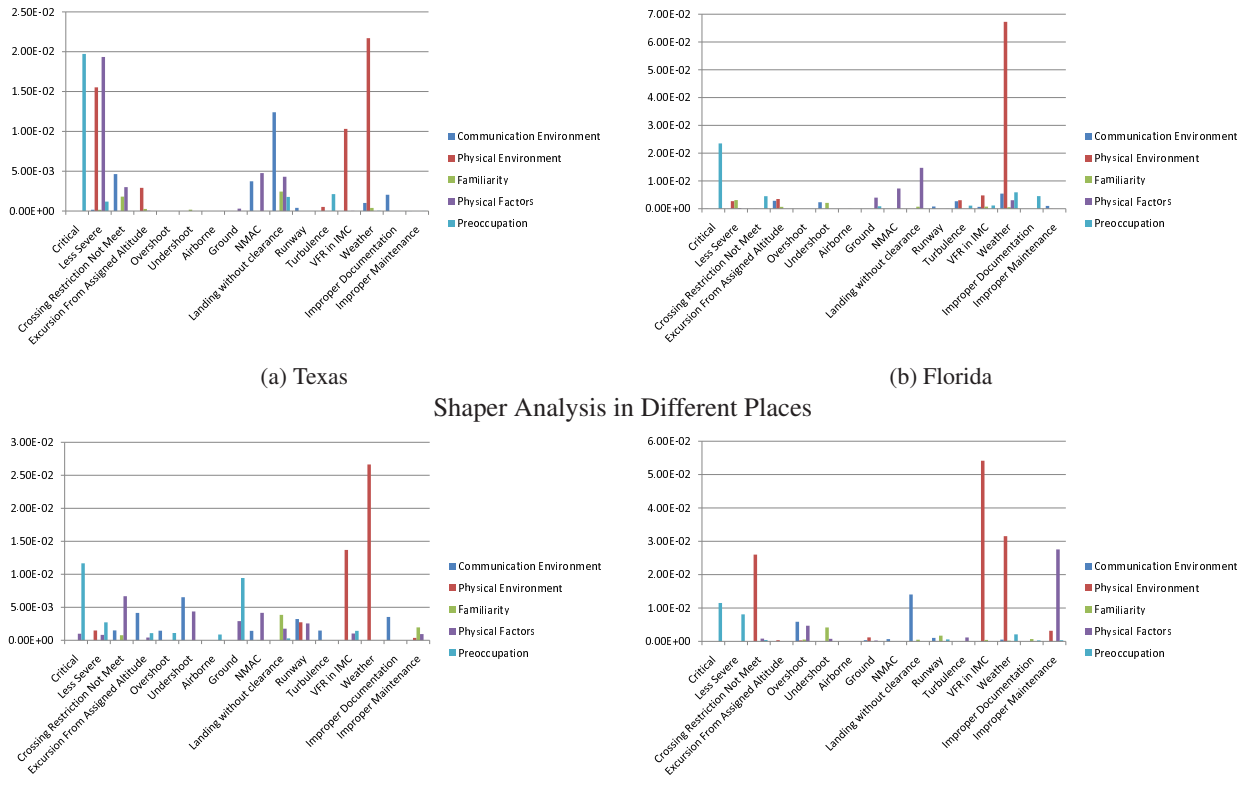


Figure 10: Comparison of Categorization Accuracy

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 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. Experiment 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 5. 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.



(a) Texas (b) Florida

Shaper Analysis in Different Places

(c) Night (d) Daylight

Shaper Analysis in Different Environment

Figure 9: Shaper Analysis in Different Context

Acknowledgment

The work was supported in part by NASA grant NNX08AC35A, the U.S. National Science Foundation grants IIS-0842769, IIS-0713571 and IIS-0713581, and the Air Force Office of Scientific Research MURI award FA9550-08-1-0265.

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

[4] S. AGARWAL, R. AGRAWAL, P. DESHPANDE, A. GUPTA, J. F. NAUGHTON, R. RAMAKRISHNAN, AND S. SARAWAGI, *On the computation of multidimensional aggregates*, in VLDB, 1996, pp. 506–521.

[5] 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, VLDB Endowment, 2007, pp. 1410–1413.

[6] D. BLEI AND J. LAFFERTY, *Correlated topic models*, in NIPS '05: Advances in Neural Information Processing Systems 18, 2005.

[7] 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.

[8] D. M. BLEI, A. Y. NG, AND M. I. JORDAN, *Latent dirichlet allocation*, Journal of Machine Learning Research, 3 (2003), pp. 993–1022.

[9] 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, VLDB Endowment, 2006, pp. 667–678.

[10] S. CHAUDHURI AND U. DAYAL, *An overview of data warehousing and olap technology*, SIGMOD Rec., 26 (1997), pp. 65–74.

[11] 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, VLDB Endowment, 2005, pp. 982–993.

[12] 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), Hong Kong, China, Aug. 2002, pp. 323–334.

[13] W. F. CODY, J. T. KREULEN, V. KRISHNA, AND W. S. SPANGLER, *The integration of business intelligence and knowledge management*, IBM Syst. J., 41 (2002), pp. 697–713.

[14] 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 (1977), pp. 1–38.

[15] 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, New York, NY, USA, 2007, ACM, pp. 905–908.

[16] 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 (1996), p. 152.

[17] T. HOFMANN, *The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data*, in IJCAI, T. Dean, ed., Morgan Kaufmann, 1999, pp. 682–687.

[18] 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, ACM Press, 1999, pp. 50–57.

[19] 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, New York, NY, USA, 2007, ACM, pp. 455–464.

[20] P. G. IPEIROTIS, A. NTOULAS, J. CHO, AND L. GRAVANO, *Modeling and managing changes in text databases*, ACM Trans. Database Syst., 32 (2007), p. 14.

[21] S. KULLBACK AND R. A. LEIBLER, *On information and sufficiency*, The Annals of Mathematical Statistics, 22 (1951), pp. 79–86.

[22] C. X. LIN, B. DING, J. HAN, F. ZHU, AND B. ZHAO, *Text cube: Computing ir measures for multidimensional text database analysis*, in ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Washington, DC, USA, 2008, IEEE Computer Society, pp. 905–910.

[23] 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, New York, NY, USA, 2008, ACM, pp. 649–660.

[24] 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.

[25] Q. MEI, C. LIU, H. SU, AND C. ZHAI, *A probabilistic approach to spatiotemporal theme pattern mining on weblogs*, in WWW, L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, eds., ACM, 2006, pp. 533–542.

[26] 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, p. 490.

[27] 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, p. 816.

[28] 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, New York, NY, USA, 2005, ACM, pp. 19–28.

[29] J. PONTE AND W. B. CROFT, *A language modeling approach to information retrieval*, in Proceedings of the ACM SIGIR'98, 1998, pp. 275–281.

[30] A. SIMITSIS, A. BAID, Y. SISMANIS, AND B. REINWALD, *Multidimensional content exploration*, Proc. VLDB Endow., 1 (2008), pp. 660–671.

[31] M. STEYVERS, P. SMYTH, M. ROSEN-ZVI, AND T. GRIFFITHS, *Probabilistic author-topic models for information discovery*, in Proceedings of KDD'04, 2004, pp. 306–315.

[32] 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, New York, NY, USA, 2006, ACM, pp. 162–169.

[33] 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, New York, NY, USA, 2008, ACM, pp. 567–580.

[34] X. WANG, C. ZHAI, X. HU, AND R. SPROAT, *Mining correlated bursty topic patterns from coordinated text streams*, in KDD, P. Berkhin, R. Caruana, and X. Wu, eds., ACM, 2007, pp. 784–793.

[35] 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, New York, NY, USA, 2007, ACM, pp. 617–628.

[36] 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, San Francisco, CA, USA, 1997, Morgan Kaufmann Publishers Inc., pp. 412–420.

[37] 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), 2001, pp. 403–410.

[38] 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, SIAM, 2009, pp. 1124–1135.