DISCOVERY DRIVEN ANALYSIS ON SEMI-STRUCTURED TEXT DATA

BY

SAMSON ABEL ROBERT HAUGUEL

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Adviser:

Associate Professor ChengXiang Zhai

ABSTRACT

Discovery Driven Analysis (DDA) is a common feature of OLAP technology to analyze structured data. In essence, DDA helps analysts to discover anomalous data by highlighting 'unexpected' values in the OLAP cube. By giving indications to the analyst on what dimensions to explore, DDA speeds up the process of discovering anomalies and their causes. However, Discovery Driven Analysis (and OLAP in general) is only applicable on structured data, such as records in databases. We propose a system to extend DDA technology to semi-structured text documents, that is, text documents with a few structured data. Our system pipeline consists of two stages: first, the text part of each document is structured around user specified dimensions, using semi-PLSA algorithm; then, we adapt DDA to these fully structured documents, thus enabling DDA on text documents. We present some applications of this system in OLAP analysis and show how scalability issues are solved. Results show that our system can handle reasonable datasets of documents, in real time, without any need for pre-computation.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1. WHAT IS OLAP?

Online Analytical Processing, or OLAP, is a technology which enables the exploration of multi-dimensional data at different level of summarization. OLAP is often used in business intelligence to support business decisions and to discover interesting patterns in corporate data.

In its simplest interpretation, OLAP is a data visualization technology, similar to the spreadsheet technology. Indeed, OLAP enables users to visualize a summary of terabytes of records, along multiple dimensions. A common application is in company sales, where the sum of all sales is displayed by region, by time or by any other specified dimension.

As in spreadsheets, one of the purposes of visualizing data is to detect any anomalous value, and then, to provide an explanation for the anomaly. OLAP technology provides very efficient support to navigate within the data, such as changing the dimensions to be displayed (e.g. display by date instead of location) or the granularity of each dimension (e.g. display the semester view instead of month view). However, OLAP does not provide support to detect anomalous values.

1.2. WHAT IS DDA?

To palliate this shortcoming, several technologies have been developed, such as Discovery Driven Analysis, or DDA. Specifically, if we view OLAP analysis as an extension of spreadsheet analysis, DDA automatically highlights the cells of the spreadsheet which have an 'interesting' or 'unexpected' value.

For instance, if an analyst is comparing the sales of different products over several months, DDA would highlight sales which are unusually high or low, compared to other sales. DDA would also suggest looking at other dimensions (e.g. store location) to explain the 'surprising' value of highlighted cells.

By giving indications to the analyst on what dimensions to explore, DDA speeds up the process of discovering anomalies and their causes. The analyst remains the major actor of the process but DDA helps her skipping the huge amount of 'common' data, and bring her directly to the 'interesting' values.

1.3. PROBLEM STATEMENT

Unfortunately, DDA can only be applied on structured data. That is, data must follow a predefined format and the value of every field (i.e. dimension) must be numerical. Thus, DDA cannot be applied on text documents. Indeed, what is the equivalent of an OLAP dimension in text documents? Moreover, given a specific document, how to compute a numerical value for each dimension?

1.4. THESIS CONTRIBUTION

We propose to overcome these difficulties with a two-stage process. First, we use a semi-supervised method, called semi-PLSA, to generate user defined text dimensions. The algorithm also computes a relevant value for each dimension, for any given document. In a second step, we adapt the DDA principles to these computed values, thus enabling DDA on text data.

In practice, in order to analyze a set of text documents (i.e. a corpus), an analyst would input a few keywords to describe each dimension she is interested in. Then, the system would structure the corpus around these dimensions and suggest interesting dimensions to explore. Applications of this process are numerous. In the simplest case, it enables OLAP analysis on text data. That is, any dataset which contains record with text fields can be the object of our method. Such dataset are increasingly common: laptop reviews with performance ratings (structured part) and user comments (unstructured part) or patient records with health assessments and doctor comments.

Another application is in exploratory text search. Since our approach allows users to define their interests (in the form of dimension), then the output of our DDA analysis helps the users to understand how the set of documents match their interests. This is closely related to the faceted search problem in Information Retrieval.

1.5. PRESENTATION LAYOUT

In chapter 2, we give a detailed overview of the two-stage process, and present a thorough account of the technology involved to extend Discovery Driven Analysis on text. In chapter 3, we discuss implementations issues and the solutions we proposed. In chapter 4, we run qualitative and quantitative experiments to test the usefulness and performance of our system. We provide a summary of related work in chapter 5 and conclude in chapter 6.

CHAPTER 2: CONCEPTS

In this chapter, we present the algorithms and concepts involved to extend DDA on text. In the first sub-chapter, we briefly review the concept of multi-dimensional text database. In the following sub-chapter, we explain how the semi-PLSA algorithm is used to generate a set of dimensions and the values of any document along each dimension. In the final sub-chapter, we introduce the details of Discovery Driven Analysis and how we adapt DDA to our specific task.

2.1. OVERVIEW OF MULTI-DIMENSIONAL TEXT DATABASE

Multidimensional databases organize data in a multidimensional structure, as opposed to the flat structure of relational databases. In a multidimensional database, the data is expressed as numerical facts, categorized by dimensions. Each fact is indexed by its dimension values and multiple dimension values can be aggregated into a single group, thus creating a new numeric fact. This aggregation operation is performed according to a hierarchy built on the dimension.

Concretely, a multidimensional database of a company typically takes sales as the numerical facts, and time and store location as dimensions. In that case, a sales analyst can explore the data through multiple views. For instance, she can query for all the sales during the first quarter, in a specific store. She can also obtain a more aggregated view of data and query for all the sales during one specific year, in the same store. This operation of moving from a detailed view to a high level view is called roll-up. The reverse operation is called drill-down.

The major advantages of multidimensional databases over relational databases are the ability to display data at different level of details, and to explore data through multiple views. In the next sub-chapters, we present how it is possible to achieve these advantages for records containing a text field.

2.2. STAGE 1: TEXT DOCUMENTS TO RECORDS

As presented in introduction, the first step of the process is to convert a set of text documents to a set of numerical vectors. Moreover, the conversion should follow user interests, expressed as keywords.

Concretely, given T user's queries (i.e. a query is a set of keywords) and a corpus of D documents, our goal is to generate D vectors of dimension T. Each dimension is associated to

a topic defined by the user's query. For instance, given a set of laptop reviews and two queries, we want to generate one topic about the first query 'laptop battery life' and another topic about the second query 'popular laptops'. If a document is highly relevant to the first topic, then the corresponding vector will have a high value for the first dimension. The same applies to other dimensions.

2.2.1. PLSA

To achieve this goal, we propose to use a popular text mining algorithm, namely the probabilistic Latent Semantic Analysis algorithm, or PLSA [7]. In essence, this algorithm extracts a given number of topics (i.e. word distributions) from the corpus, and expresses each document as a mixture of these topics. That is, given a corpus of documents on laptop reviews and a number of topics T, PLSA will output T word distributions, where one distribution, for instance, gives a high probability to the words 'battery' 'life' and 'long'. In that context, a single document will be converted, for example, to a vector (30% topic1, 20% topic2, 50% topic3).

Formally, PLSA proposes a generative model of text documents as described in Fig 2.1. In PLSA framework, a document is generated by choosing a mixture of topics θ (e.g. 10% of topic1, 80% of topic 2 and 10% of topic 3), then picking a topic z according to this mixture and finally, generating a word w according to this topic z (i.e. word distribution). The process of choosing a topic z is repeated for every N words to be generated in the document. State of the art PLSA algorithms assume that there exists a background topic, in addition to regular topics. The purpose of this special topic is to gather all non-discriminative and non-informative words, such as 'the', in order to maximize the relevance of regular topics. Therefore, the probability $p_d(w)$ of generating a word w, for a specific document d, is summarized in equation (1).

$$p_d(w) = \lambda_B p(w|\theta_B) + (1 - \lambda_B) \sum_{t=1}^{T} (\pi_{d,t} p(w|\theta_t)) \quad (1)$$

Where:

$p(w|\theta_B)$ is the word distribution for the background topic θ_B

$p(w|\theta_t)$ is the word distribution for the regular topic θ_t

λ_B is the probability of picking topic θ_B

$(1 - \lambda_B)\, \pi_{d,t}$ is the probability of picking topic θ_t

If we assume this model for our corpus, then the goal is to 'guess' what the topics and mixtures are, given the observations of documents and word. The topic mixtures $\pi_{d,t}$ are exactly the documents' vector representation we are looking for. Formally, if we assume that word position in documents does not matter (i.e. bag-of-words assumption), our goal is to maximize the log-likelihood of our corpus D, given in the following equation:

$$\log p(D|\Lambda) = \sum_{d\epsilon D}\sum_{w\epsilon V} c(w,d)\log\left(\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{t=1}^{T}(\pi_{d,t}p(w|\theta_t))\right) \quad (2)$$

Where:

$\Lambda = \{\theta_B, \pi_{d,t},\ \theta_t \mid \text{d}\epsilon\text{D},\ 1 \leq \text{t} \leq \text{T}\}$ is the set of maximizing variables

$c(w,d)$ is the function which returns the count of word w in document d

The optimization problem can be stated as:

$$\widehat{\Lambda} = \underset{\Lambda}{\operatorname{argmax}} \log p(D|\Lambda) \quad (3)$$

One common way to solve this optimization problem is to use the Expectation-Maximization, or EM, algorithm. The EM algorithm is an iterative process, where the maximizing variables are estimated according to the following updating formulas:

$$p(z_{d,w,t}) = \frac{(1-\lambda_B)\,\pi_{d,t,n}p(w|\theta_t)_n}{\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{t=1}^{T}(\pi_{d,t,n}p(w|\theta_t)_n)} \quad (4)$$

$$p(z_{d,w,B}) = \frac{\lambda_B p(w|\theta_B)}{\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{t=1}^{T}(\pi_{d,t,n}p(w|\theta_t)_n)} \quad (5)$$

$$\pi_{d,t,n+1} = \frac{\sum_{w\epsilon V} c(w,d)p(z_{d,w,t})}{\sum_{t\epsilon T}\sum_{w\epsilon V} c(w,d)p(z_{d,w,t})} \quad (6)$$

$$p(w|\theta_t)_{n+1} = \frac{\sum_{d\epsilon D} c(w,d)p(z_{d,w,t})}{\sum_{w\epsilon V}\sum_{d\epsilon D} c(w,d)p(z_{d,w,t})} \quad (7)$$

Intuitively, these formulas tell us that if two words (e.g. 'battery' and 'life') co-occur frequently, then if one word has a high probability in the topic, then the other should also

have a high probability. Thus, PLSA formalizes the notion of word co-occurrence and clusters similar words in the same topic.

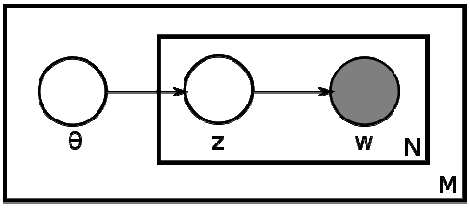


Figure 2.1: Plate notation for PLSA.

2.2.2. Semi-PLSA

Although meaningful, extracted PLSA topics θ_t may not match user interests. Indeed, PLSA does not take any user input, beside the number of topics. In order to integrate users priors (i.e. how the topics should look like), we used the semi-PLSA algorithm [9]. Intuitively, this semi-PLSA algorithm starts from given word distributions and modifies them to obtain the final topics.

Formally, given a user prior $p(w|u_t)$ for each topic and a confidence in these priors σ_t, we define the prior (or bias) for all parameters by the following equation:

$$p(\Lambda) \propto \prod_{t=1}^{T} \prod_{w \epsilon V} p(w|\theta_t)^{\sigma_t p(w|u_t)} \tag{8}$$

This equation is a rigorous way of integrating user's priors as training data, when estimating the new topics. Hence, we define the corresponding optimization problem:

$$\widehat{\Lambda} = \underset{\Lambda}{\operatorname{argmax}}\, p(D|\Lambda)\, p(\Lambda) \tag{9}$$

This problem can be solved by using a slightly different variation of the EM algorithm, as shown by the following formulas.

$$p(w|\theta_t)_{n+1} = \frac{\sum_{d \epsilon D} c(w,d) p(z_{d,w,t}) + \sigma_t p(w|u_t)}{\sum_{w \epsilon V} \sum_{d \epsilon D} c(w,d) p(z_{d,w,t}) + \sigma_t} \tag{10}$$

In essence, these formulas express that each topic is estimated by the count of words seen in each document $c(w,d)$ and a pseudo count of words given by user priors $\sigma_t p(w|u_t)$. In

this setting, σ_t is a parameter tuning the confidence in each user prior, that is, how many pseudo counts we should add for a specific word.

2.2.3. User Priors

Semi-PLSA provides a principle way to integrate user's interests into the topic generation process. However, it requires the priors to be expressed as a word distribution over a vocabulary of thousands of words. That means the users are expected to input a probability for each word, in order to generate a prior. One simple solution is to give equal probability weights for the few keywords the users are willing to give, and a zero weight for all other words.

However, this solution leads to poor results, because the priors are not 'rich' enough. Hence, to convert the few user keywords to a rich set of user priors, we propose the following approach, as explained in [8].

Given a set of documents, we first build a directed graph of similar words. That is, each node of the graph is a word, and each weighted, directed edge carries the similarity between a pair of words. The similarity function, in equation (12), is based on the mutual information measure of two words, as in equation (11). Essentially, if two words co-occur together frequently, then they will be judged similar.

Once the graph is built, we propagate the probability weight of the keywords nodes (i.e. words chosen by users) to its nearest neighbors. This iterative process is expressed by equation (13), which shows how the user's prior for one topic is updated.

Point-wise Mutual Information:

$$MI(i,j) = log\frac{p(w_i, w_j)}{p(w_i)p(w_j)} \tag{11}$$

Where:

$p(w_i, w_j)$ is the co-occurrence probability of words i and j

$p(w_i)$ is the occurrence probability of word i

Similarity function of the MI graph, for directed edge $\overrightarrow{w'w}$:

$$sim(w,w') = \frac{MI(w,w')}{\sum_{w''\in V} MI(w',w'')} \tag{12}$$

Where:

w, w', w'' are words and nodes from the vocabulary

Propagation formula for topic θ_t:

$$p(w|\theta_t)_{n+1} = (1-\lambda)p(w|\theta_t)_0 + \lambda \sum_{w' \in V} sim(w,w')p(w'|\theta_t)_n \quad (13)$$

Where:

$p(w|\theta_t)_0$ is the initial word probability of node w

$sim(w,w')p(w'|\theta_t)_n$ is the probability mass which is transferred from word w' to word w

λ is a parameter balancing the amount of transferred probability mass

We present in chapter 3 a few approximations of this process that we used to speed up the computation.

2.3. STAGE 2: EXTENSION OF DDA

In this subchapter, we present how we apply Discovery Driven Analysis on the output obtained in previous paragraphs. Our main objective is to help users discovering 'unexpected' data from the corpus. We first define a model of interestingness of the data. We then derive a set of interestingness metrics, from this model. The challenges associated with the computation of these metrics are not discussed until chapter 3.

2.3.1. Model of interestingness

In order to detect 'unexpected' values, we must first define what it is an 'unexpected' value. Typically, a DDA system computes an expected value for each cell of the OLAP cube. If the normalized difference between the expected cell value and its actual value is more than a pre-set threshold, then the cell value is considered 'unexpected'. This idea is summarize by equation (14).

$$s_{i_1,i_2,\ldots,i_T} = \frac{|y_{i_1,i_2,\ldots,i_T} - \hat{y}_{i_1,i_2,\ldots,i_T}|}{\sigma_{i_1,i_2,\ldots,i_T}} \quad (14)$$

Where:

$i_1, i_2, \ldots, i_T$ are the coordinates of the cell which value for dimension 1 is i_1, etc.

$s_{i_1,i_2,\ldots,i_T}$ is the surprise of the cell

$y_{i_1,i_2,\ldots,i_T}$ is the actual value of the cell

$\hat{y}_{i_1,i_2,\ldots,i_T}$ is the expected value of the cell

$\sigma_{i_1,i_2,\ldots,i_T}$ is the standard deviation of the residual $|y_{i_1,i_2,\ldots,i_T} - \hat{y}_{i_1,i_2,\ldots,i_T}|$

Therefore, the problem is translated into: how do we compute an expected value $\hat{y}_{i_1,i_2,\ldots,i_T}$ for any cell?

Answering this question is one of the main tasks of any DDA system. Depending on the assumptions we make about the users, the output of a DDA process may vary a lot. For example, the first DDA implementation ([16]) stated that the expected value for a cell depends on the average values of higher level cells. Intuitively, it means a user expects to see values that correspond to higher level (more summarized) cells. For example, let us consider the sales OLAP cube of a company. If the average sales for all the products and for year 2009 is $X, then, we expect the average sales for a specific product to be around that average.

Other models are possible, which capture different intuitions about what a user expects. Before introducing our own model, we first start with two definitions.

Regular dimension and topic dimension:

The process explained in chapter 2 is to structure text document along several new dimensions. These dimensions are called topic dimension. However, we are focusing on text documents that come as a text field from a database record. That is, text documents already have some structured data appended. For example, a laptop review may have a date field, a rating field etc. In this case, we define the pre-existing fields as regular dimensions.

The purpose of this distinction is to provide some guidance for the analysis. Simply said, regular dimensions often integrates factual data, which can be seen as causes, whereas topic dimensions are computed from text documents, and can be assimilated to observations or consequences. That is, when the value for a regular dimension changes, users are usually interested in observing how the text changes. The underlying justification is that text data often appears as a description of the structured data. Of course, this intuition may not be accurate in some cases.

Given a set of regular and topic dimensions for a specific corpus, our interestingness model states that the values follow the same distribution across different values of a regular dimension. For instance, if we consider a regular dimension such as Location, then the values distribution for the topic dimension T1 are expected to be the same for Location = America, Location = Asia and Location = Europe.

Intuitively, if a user does not have any prior about her data, then she expects all the regular dimensions value to exhibit the same topic profile. That is, every text document is composed from the same mixture of topics. In case when one or more documents significantly differs from this 'average profile', our model suggests that these documents are interesting.

2.3.2. Measure of interestingness

In this paragraph, we focus on adapting our model of interestingness into concrete metrics. As stated earlier, after stage 1 is performed, all text data are expressed as multidimensional vectors. Some dimensions are regular, meaning that the value for these dimensions come from the structured part of data. The other dimensions, called topic dimensions, are generated by the PLSA algorithm.

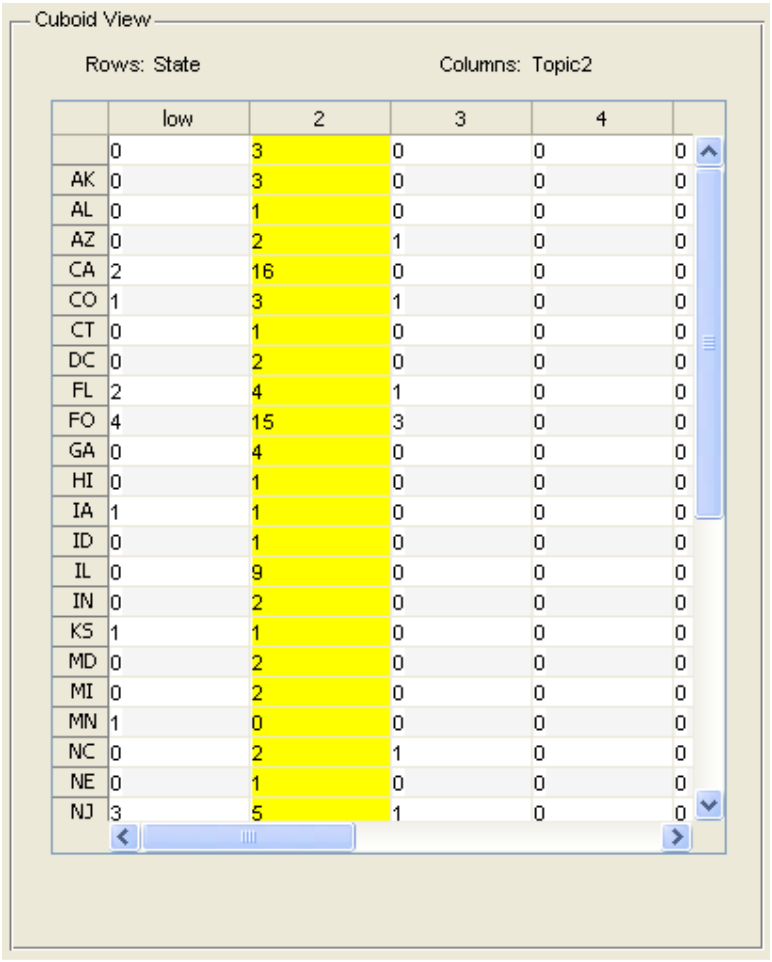
Our model assumes that for a given regular dimension, if we group documents according to their regular dimension value, then every group should exhibit a similar topic distribution. Mathematically, this is formalized by the following definitions. Let D by a corpus of semi structured text documents, that is, free-form text documents augmented with structured fields. Let R be the set of regular dimensions, and T the set of topic dimensions. The set of topic dimensions T depends on the user's queries. For any dimension d in $R \cup T$, we will denote its possible values (i.e. domain) by $d_1, d_2, \dots d_{|d|}$.

The model says that for a given regular dimension r in R, and for any given topic dimension t in T, the value distribution for T is similar across different document groups. The groups are defined by having the same r_i value (i.e. group by R statement in SQL).

We choose to define group similarity in terms of normal distribution. Specifically, if we consider each group as a vector along the T dimension, then we state that each of these vectors are drawn from the same Gaussian distribution.

For example, let r be a 'Location: State' regular dimension and t be a 'Topic1: Relevance' topic dimension. The values $r_1, r_2, \dots, r_{|r|}$ of the regular dimensions could be renamed as 'Illinois', 'Indiana' and so on. Similarly, the values $t_1, t_2, \dots, t_{|t|}$ of the topic dimensions could be ordered and renamed as 'low relevance', 'medium relevance' and so on. A group of document could be all documents such that r value is 'Illinois'. Such group has a specific topic values distribution. For instance, 20% of documents have a value of t_1 for 'Topic1', 5% have a value of t_2 for 'Topic1', etc. We can express this as a vector $v_{Illinois}$ containing every percentage. The assumption of our model states that every v_{r_i} is drawn from the same normal distribution.

Fig. 2.2 explains the same concept from a tabular point of view. Each row expresses a specific value for the regular dimension 'Location: State'. Each column expresses a specific value for a topic dimension 'Topic2: Relevance'. The value in each cell is the count of every document matching both dimension values. In this figure, every vector, such as $v_{Illinois}$, is mapped to a row of the table.


Cuboid View

Rows: State Columns: Topic2

	low	2	3	4	
	0	3	0	0	0
AK	0	3	0	0	0
AL	0	1	0	0	0
AZ	0	2	1	0	0
CA	2	16	0	0	0
CO	1	3	1	0	0
CT	0	1	0	0	0
DC	0	2	0	0	0
FL	2	4	1	0	0
FO	4	15	3	0	0
GA	0	4	0	0	0
HI	0	1	0	0	0
IA	1	1	0	0	0
ID	0	1	0	0	0
IL	0	9	0	0	0
IN	0	2	0	0	0
KS	1	1	0	0	0
MD	0	2	0	0	0
MI	0	2	0	0	0
MN	1	0	0	0	0
NC	0	2	1	0	0
NE	0	1	0	0	0
NJ	3	5	1	0	0

Figure 2.2: View of Cuboid ('Location:State', 'Topic2:Relevance').

This is one of the possible implementations of our interestingness model. Its simplicity allows defining concrete metrics.

2.4. METRICS FOR DDA

2.4.1. Metric 1: Cells Comparison

We first estimate the parameters of the normal distribution, using the simple mean and variance estimators. Then we discretize the obtained distribution into buckets, or intervals, of equal lengths. Length of intervals should be adapted to the measure used in the data cube. In

our case, we selected count_as_percentage() as our measure for each cell. Thus, the values of each cell (i.e. the value of the group vector) must be an integer between 0 and 100, and a precision of plus or minus 0.5 point was deemed sufficient. This translated into intervals of length 1, centered on integer values (e.g. interval is 3.5 to 4.5, for instance). Once the distribution is discretized, every interval is assigned a probability, according to the parameter of the normal distribution. Eventually, every data point is assigned a probability, which is the same as its bucket probability. The pseudo code to compute the probability for each data point is given below.

```
// Surprise of each cell
Double[][] Prob;

// for each column
for col =1: numColumn,
   // get all the rows
    values = getRows(col)

   //estimate parameters of normal distribution for the column
   mu = mean(values);
   sigma = std(values);

   // for each cell in the column
   for row = 1:numRow,

      //compute probabilities, assuming normal distribution
      //note: probability of interval [x +- 0.5]
      Prob(row,col) = normcdf(values+0.5,mu,sigma) - normcdf(values-0.5,mu,sigma);

      //normcdf(x,m,s) computes the cumulative distribution function at point x,
      // of a Gaussian with mean m and standard deviation s.
   end
end
```

Figure 2.3: Pseudo-code to compute the surprise of each cell.

With this ability of rating each data point, each component of a group vector can be rated and compared to the same component of another group vector. For instance, let us assume that $v_{Illinois}$ first component value for topic 'Topic1' is 20%, as in the previous example. This value may be given a probability of 0.65. This probability means that 20% is a common value amongst all the first components of every group vector.

For our system, the final metric is based on fixed thresholds. Specifically, if a value has a probability less than 5%, it is classified as surprising; if less than 1%, it is classified as very surprising. This metric is summarized in equation Figure 2.4.

We showed the result of applying this metric in figure 2.2. Different colors are associated to different thresholds.

cellCompare(cell) = white	5% < Prob(cell)
cellCompare(cell) = yellow	1% < Prob(cell) < 5%
cellCompare(cell) = red	Prob(cell) < 1%

Figure 2.4: Cell comparison metric.

Metric 1 purpose is to help analysts to detect unusual data points, at the cells level. Therefore, this metric gives a fine granularity for analyzing regular dimension values and their impact on the topic dimension values. However, analysts are also interested in more aggregated view of their data. Metric 2 is intended to provide a higher level view of the data, by comparing dimensions.

2.4.2. Metric 2: Dimension Comparison

The first intuition to build an aggregated metric is to combine lower level measures into a single high-level measure. For example, given all the cells' interestingness for a specific cuboid, it is possible to rate the cuboid itself by averaging all the cells' interestingness. Since cuboids are expressed as a set of dimensions, then this cuboid level rating can be converted to a dimension level rating, as we will show later.

However this simple view does not take into account the different semantics of dimensions. Indeed, regular dimensions and topic dimensions are different class of dimensions. Our model assumes that topic values are expected to be similar across regular dimensions values, but no assumption is made on topic values within one topic dimension. Simply said, our model expects that the group vectors $v_{Illinois}$ (as introduced in previous examples) and $v_{Indiana}$ will be similar, but nothing is stated about the components of $v_{Illinois}$.

To palliate this problem, our dimension comparison metric will have a different interpretation whether applied on a regular dimension, or on a topic dimension. Specifically, the goal of this metric is to answer the question 'what is the most interesting topic?' when applied on a regular dimension. Similarly, the metric answers the question 'what is the most interesting regular dimension?' when applied on a topic dimension. In practice, this translates into an advice to explore (i.e. drill down) a dimension in order to obtain a pair regular dimension - topic dimension. Such pairs are desirable because our framework assimilates regular dimensions to causes and topic dimensions to consequences.

With these distinctions in mind, the implementation of a dimension comparison metric is straightforward. We proposed a metric which takes as input the currently viewed cuboid and returns a rating for every dimension. The idea is to compute an interestingness score for all the cuboids reachable from the current cuboid, by expanding one dimension. For example, if a user is viewing the one dimensional cuboid (State,*,*), where the other dimensions 'Topic1' and 'Topic2' are aggregated, then the system will compute an interestingness score for the 2 cuboids (State, Topic1, *) and (State, *, Topic2). The cuboid (State, Topic1, Topic2) is not computed because it cannot be reached from the initial cuboid by expanding only one dimension. Since only one dimension is expanded at the time, the score of each dimension is defined by the score of the cuboid obtained.

In order to avoid mixing regular dimension scores with topic dimension scores, only one class of dimensions is expanded. For example, when the current cuboid is viewed along a regular dimension, only topic dimensions are expanded in the computation of dimension scores. If the current cuboid is viewed along a topic dimension, regular dimensions will be expanded instead. Pseudo code for computing this metric is given in Fig. 2.5.

```
// Score of each dimension
Double[] Score;

// expand topic dimension if current dimension is regular; and vice versa
// expand regular dimension if no current dimension is selected
for dim in ComplementDim

    //generate the 1D or 2D cuboid, by expanding dim
    Cuboid = expandCuboid(currentCuboid, dim)

    //compute surprise of each cell
    Prob = genProb(Cuboid)

    //minimum of all cells surprise value
    Score(dim) = min(Prob)
end
```

Figure 2.5: Pseudo-code to compute the score of each dimension.

Note that computing cuboid interestingness is very flexible. It is mostly a matter of aggregating the cells measure obtained with metric 1. In our case, the aggregation function is the minimum measure among all cells, as describe in Fig. 2.5. The justification is that our target analyst is interested in finding very unusual data point. Of course, other aggregation functions, such as the average, can be used instead of minimum.

The computation of dimension scores highlights a specificity of our system: cuboids with more than 2 dimensions are not explicitly supported. Put it another way, once the data cube has been expanded along one regular dimension and one topic dimension, the system does not allow more expansion. This is similar to spreadsheets where only slices of 2D data can be viewed. This does not mean that some cells are not reachable, because the analyst is always free to specify a value for any dimension (e.g. State = 'Illinois'). The whole data cube is accessible through slices of 2 dimensions. However, there is no support for expanding dimensions and viewing cuboids with more than 2 dimensions. In practice, this is not a problem since humans can hardly digest data with more than 2 dimensions.

Overall the system achieve its goal of providing a high level understanding of data by guiding analysts at the dimension level and at the cell level. A complete view of our system in usage is given in Fig. 2.6. The upper left part takes users queries; the central part shows a view of the cuboid and the right part show the dimensions (i.e. a view of the datacube).

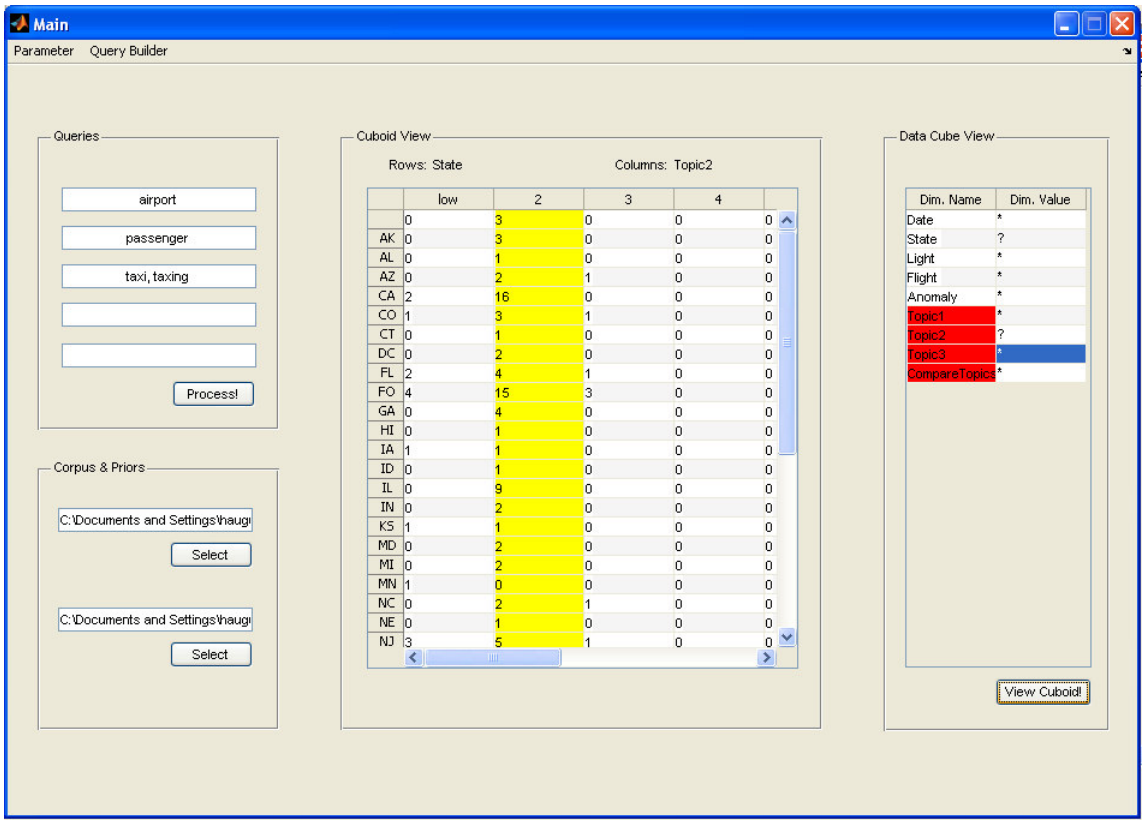


Figure 2.6: Complete View of the system.

2.5. ADDITIONAL FEATURE: TOPIC COMPARISON

As a convenience, our system provides extended support to compare topic dimensions. In addition to the dimension comparison metrics, the system is also setup to compute the average topic values for every topic, and present all these values in a single view.

For instance, given a regular dimension such as 'Location:State', a special dimension, called 'TopicCompare', is available for expanding. This extra dimension has the topic dimensions as values, meaning that each state from the 'Location' dimension is broken down by topics. Therefore, analysts will see the average value for Topic1 and Topic2, for Location=Illinois. The same applies for any other Location.

The special dimension 'TopicCompare' is also considered as a topic dimension for the purpose of cell comparison. Hence, cells will be highlighted within the cuboid (regular dimension, 'TopicCompare') indicating surprising values.

The advantage of such feature over the regular dimension metrics is to provide a finer granularity to compare all topics. While the dimension comparison metrics answer questions such as 'What is the most interesting topic dimension for the location dimension?', this feature intends to answer questions such as 'What are the most interesting topic dimensions for location = Illinois?'.

Of course, combining the dimension comparison and cell comparison metrics achieve the same result as this feature. An analyst could bypass the 'TopicCompare' dimension and compare the topics manually by expanding all the topic dimensions successively, and noting down the results of comparing cells. Obviously, this adds a burden on the user and should be avoided if possible.

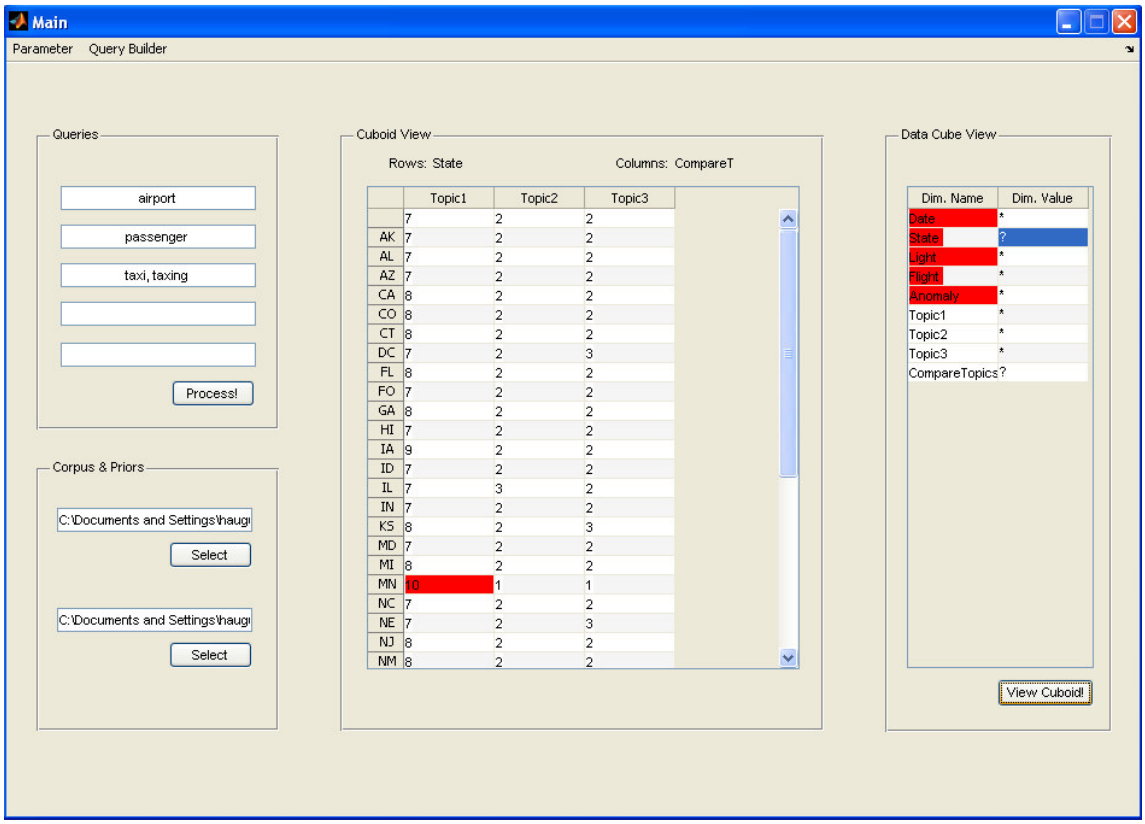


Figure 2.7: View of cuboid (‘Location:State’, ‘CompareTopic’).

CHAPTER 3: TECHNICAL CHALLENGES

This chapter deals with the implementation issues of our process. Concretely, we explain how we generate user priors efficiently for the semi-PLSA algorithm (first stage of process). In a second subchapter, we present our technique to efficiently compute and store quantities involved in the DDA process.

3.1. APPROXIMATION OF USER PRIORS

As presented earlier, one of the major phases of stage 1 is to expand the few user keywords into a richer word distribution. This is done by generating the Mutual Information graph and then propagating the weight of the initial keywords to their surrounding neighbors. We restate in equations (15) and (16) how to compute the Mutual Information and how to obtain the Mutual Information graph.

$$MI(i,j) = log\frac{p(w_i, w_j)}{p(w_i)p(w_j)} \tag{15}$$

$$sim(w,w') = \frac{MI(w,w')}{\sum_{w''\in V} MI(w',w'')} \tag{16}$$

Assuming D documents in a corpus and V different words in the vocabulary, it is easy to see that computing the Mutual Information is proportional to D (it is just a matter of counting the co-occurrences of words). Since the MI graph contains V nodes and is complete, then the whole computation of the MI graph is proportional to DV^2. For D=1000 documents and V=10,000 words the computation takes days on a personal computer (as of year 2010). Besides, the MI graph has a size in the order of V^2 doubles, which means around 800 MB of memory (assuming 8 bytes per double). While this data size is manageable for current computers, it still requires a lot of system tuning and a good memory management.

To palliate this high computation cost, our system uses an approximate algorithm to expand user keywords. The main idea of this approximate algorithm is to compute a partial MI graph, centered on the keywords node. This technique helps reducing the space cost and computation cost, as we will show in the experimental chapter (see Chapter 4). The second optimization is to perform a single pass propagation from the keyword nodes to their neighbor nodes. This idea helps reducing the computation cost in two ways: the propagation is performed only once and only from the keywords nodes to their neighbors. By comparison, the original algorithm does several iterations, and each iteration propagates the weight of every node to their neighbors. The pseudo code for the approximate algorithm is given in Fig. 3.1.

```
// PART1: partial computation of MI graph
//compute only the edge weight between users keywords and other words
Double[][] TransferWeights

//for each keyword
for kw = 1:keywords,
   //get Mutual Information between keyword and any word
   for w = 1:words,
      TransferWeight(kw,w)  = mi(kw,w,dataset);
   end
  //normalize: all edges going out from one node sum to 1
  TransferWeights = Norm(TransferWeights);
end

// PART2: Propagate Probability Mass of keywords (first neighbors only)
// Final word distribution is initialized with user input Pw_u0
Double[] Pw_u = Pw_u0;
// Keyword word distribution is initialized with user input Pw_u0
Double[] Pkw_u = Pw_u0;

// for each iteration of the process
for ite=1:ites,

   // Compute the probability mass transferred from keywords to neighbors
   // (weighted sum)
   Pw_transfer1 = sum(TransferWeights * diag(Pkw_u),2);

   // Compute the probability mass transferred from nodes to themselves
   // (copy from previous iteration)
   Pw_transfer2 = Pw_u .* NonKeywordInd;

   // Propagate, according to parameter lambda
   Pw_u = (1-lambda)*Pw_u0 + lambda * (Pw_transfer1 + Pw_transfer2);

   // Update Pkw_u
   Pkw_u = Pw_u(KeywordInd);
end
```

Figure 3.1: Pseudo code of the fast keyword expansion algorithm.

3.2. APPROXIMATION OF SEMI-PLSA

The second major phase in stage 1 is to perform the semi-PLSA algorithm. As show in chapter 2, semi-PLSA algorithm is very similar to the standard EM algorithm. As such, it performs multiple iterations until the likelihood stabilizes, and each iteration has a computation cost proportional in the number of documents (see equation 17).

$$cost_{EM-step} \propto TVD \quad (17)$$

Where:

T is the number of topics

V is the size of the vocabulary

D is the number of documents

Again, for D=1000 documents and V=10,000 words the computation takes hours on a personal computer. To palliate this high computation cost, our system resort to an approximation of PLSA algorithm. Specifically, our system performs a standard semi-PLSA algorithm on a random subset of the original corpus. The result of this computation, that is, the topic mixture for each sampled document, is propagated to the rest of the corpus, in similar way to the keyword weight propagation described above.

The idea is to build a similarity graph, where each node is a document and each edge bears a weight corresponding to the similarity between documents. We used the cosine function as our similarity function. Then, every sampled document is associated with the result of the PLSA computation. Finally, every remaining node has its topic dimension values computed by performing a weighted average of the sampled documents' topic values. Equations (18) and (19) summarize this last step.

$$p(z|d_{notSampled}) = \frac{1}{Z} \sum_{d_s \in D_{sampled}} sim(d_{notSampled}, d_s) p(z|d_s) \quad (18)$$

$$sim(d_{notSampled}, d_s) = \frac{cosine(d_{notSampled}, d_s)}{\sum_{d \in D_{notSampled}} cosine(d, d_s)} \quad (19)$$

Where:

Z is a normalization factor

$D_{sampled}$ is the set of sampled documents

$D_{notSampled}$ is the set of not sampled documents

Notice that propagating results is a single pass operation, and requires only the similarity between sampled documents and remaining documents. The soundness and effectiveness of this approximate algorithm is discussed in the experimental chapter. Intuitively, this algorithm speeds up the computation by trading a complex operation (i.e. PLSA on all documents) for

two simpler operations (i.e. PLSA on subset of documents and propagating results). However, the tradeoff is a loss in accuracy for the computation of topic values.

3.3. FAST COMPUTATION OF DDA INDICATORS

While the computation of DDA indicators present no technical difficulty, it is still interesting to study why the system does not encounter an 'explosive computation cost', which is a common challenge in data cube systems. As a matter of fact, computations involving cuboids, such as the dimension comparison metrics, often have to face a combinatorial growth in the number of cuboids. Fortunately, some features of our system prevent this combinatorial growth and thus allowing the computation of DDA indicators online. In this paragraph, we study the system's features that help keeping the computation cost low.

3.3.1. 2D Cuboids Only

Cuboids which will be generated have at most 2 dimensions. As stated earlier, users are free to visualize any 2D slice of larger cuboids. However in these cases, the system does not generate the whole cuboid but only the required slice. This constraint does not impair the data visualization abilities of our system, nor the DDA indicator. Indeed, DDA indicators are built on the data a user can visualize, and since the data is at most 2-dimensional, the size of the data is always bounded by the maximum number of values in any pair of dimensions. This is summarized by equation (20):

$$cuboidSize < \max_{\mathrm{dim}_1,\mathrm{dim}_2 \in R \cup T} |dim_1| * |\,\mathrm{dim}_2\,| \qquad (20)$$

Where:

T is the set of topic dimensions

R is the set of regular dimensions

|dim| is the range of dimension dim

3.3.2. Computing Scores of Reachable Cuboids Only

Even with the 2D cuboids restriction, the computation of every 2D cuboid and 2D slice, and their corresponding DDA metrics, costs too much time and space. However users are interested only in the cuboids reachable from the cuboid currently visualized. As such, the system needs only to compute a score for every neighbor cuboids (i.e. one drill down away

from the current cuboid). These are limited by the number of dimensions, as seen in equation (21).

Besides, slice suggestion is taken care of by the cell suggestion metrics. If a user observes an interesting cell value, she performs a slicing operation by constraining one or both dimensions to their respective dimension value. In effect, this reduces the visualization to a 1D or 0D cuboid. Afterwards, the dimension suggestion module proposes new dimensions to explore. Overall, for any 2D cuboids or 2D slices, the system needs only to score the neighboring cuboids.

$$neighboringCuboids < |R \cup T| \tag{21}$$

Where:

$|R \cup T|$ is the number of dimensions

3.3.3. Distinction Between Topic Dimensions and Regular Dimensions

This last specificity speeds up the computation a little, by effectively reducing the number of 'interesting' cuboids. Specifically, a user is recommended to view 2D cuboids where one dimension is regular and the other is topical. This is not an obligation, and a user is perfectly allowed to visualize a topic by topic, or regular by regular cuboid. However, the DDA indicators are computed only for these interesting 2D cuboids, where the two dimensions are from different classes.

This 3rd feature reduces the number of neighbors to generate, as expressed in equation (22).

$$neighboringCuboids < \max(|R|, |T|) \tag{22}$$

All these features allow a straight forward computation of DDA indicators. The pseudo code for this computation was given in Fig 2.5.

This chapter introduced some of the technical challenges related to the implementation of DDA on text. The next chapter will focus on testing the system.

CHAPTER 4: EXPERIMENTS

In this chapter, we propose several experiments to test the validity of our system. Especially, we focus on testing the merit of our approximate algorithms, presented in chapter 3. Consequently, the first subchapter will discuss the tradeoff scalability for accuracy involved with these approximate algorithms. The second subchapter will present some typical uses of the system, in order to prove it usefulness.

4.1. SCALABILITY AND ACCURACY

As presented in the previous chapter, the system relies on several approximations to speed-up the computation of user priors and of PLSA algorithm. However, these speed-ups come at the cost of a loss of accuracy. Therefore, our goal is to tune the parameters of these approximate algorithms in order to obtain a good speed-up while maintaining a reasonable accuracy.

4.1.1. Approximation of user priors

As shown in Fig. 3.1, the major difference with the original algorithm (see [8]) is the propagation of probability mass from keywords to neighbors only. Indeed, the original algorithm propagates the probability mass from each word to all of its neighbors, which necessitates the computation of the whole Mutual Information graph. However, the goal of this algorithm is not so much about computing the relevance of every word, with regard to the initial keywords. It is rather about finding the most relevant words, related to these initial keywords.

To measure the quality of the approximation, we assume the results of the original algorithm to be our goal standard. Specifically, each word of the final word distribution (i.e. expanded user prior) is ranked by its probability mass. Since words in the final approximate word distribution can also be ranked by their probability mass, we reduce the initial problem of measuring accuracy, to the more common problem of comparing two ranked lists. That is, if we define a probability mass threshold for a word to be relevant, we enable the use of two typical Information Retrieval measures, namely precision and recall. Formally, given the original ranked words list, the last rank r_{max} for a word to be relevant is such that:

$$\sum_{r=1}^{r_{max}} p(w_r|\theta_{orig}) = 80\% \tag{23}$$

Where:

w_r is the word of rank r, in the original word distribution

$p(w_r|\theta_{orig})$ is the probability mass assigned to that word

80% is an arbitrary threshold

Since the set of relevant words is defined, and given that the approximate algorithm retrieves only a subset of words (i.e. only a subset of words have a non-zero probability), precision and recall can be defined for the approximate algorithm.

$$precision = \frac{|relevant \cap retrieved|}{|retrieved|} \tag{24}$$

$$recall = \frac{|relevant \cap retrieved|}{|relevant|} \tag{25}$$

Fig 4.1 shows the top 15 words in the original and the approximate algorithm (they are the same), for the query 'takeoff'. Fig 4.2 presents the precision and recall of the approximate algorithm for multiple queries.

Word	Original Prob.	Approx. Prob.
takeoff'	0.10118	0.10007
time'	0.00152	0.00069
did'	0.00139	0.00069
landing'	0.00139	0.00069
continued'	0.00132	0.00069
departure'	0.00128	0.00069
aircraft'	0.00125	0.00069
called'	0.00125	0.00069
asked'	0.00124	0.00069
runway'	0.00121	0.00069
flight'	0.00119	0.00069
right'	0.00116	0.00069
possible'	0.00116	0.00069
just'	0.00116	0.00069

Fig 4.1: Expanded words from query 'takeoff', and their probability.

# queries	100	20	20	10
# words in V	3013	4053	5037	6012
# docs in D	97	124	163	205
avg precision	0.995	0.998	0.992	0.994
avg recall	0.025	0.019	0.021	0.013

Fig 4.2: Precision and recall for approximate algorithm

The precision of 99% on average is easy to understand. Indeed, both algorithms rank the first neighbors of the initial keywords very high. However, the approximate algorithm does

not propagate the probability mass further. That is, a lot of words which are related loosely to the initial keyword do not receive any mass, with the approximate algorithm. This is why the recall is very low. The tradeoff for this poor recall is a good speed-up of x3000 on average. The speed-ups obtained for different vocabulary size are reported in Fig 4.3.

# queries	100	20	20	10
# words in V	3013	4053	5037	6012
# docs in D	97	124	163	205
Original algo. avg run time	56	124	251	434
incl. Transfer Weights	53	112	243	420
Approx. algo. avg run time	0.027	0.049	0.072	0.09
incl. partial Transfer Weights	0.02	0.04	0.06	0.08
Speed-up	2074	2531	3486	4822

Fig 4.3: Average running time (in seconds) and speed-up for approximate algorithm.

As seen from Fig 4.3, the major computation gain is obtained by computing only a few edges from the M.I. graph. Indeed, the computation of the whole M.I. graph is in the order of the square of the vocabulary size V (i.e. $O(V^2)$) while the approximate algorithm computes only the edges from the k keywords to their V neighbors (i.e. $O(kV)$ $where$ $k \ll V$).

4.1.2. Approximation of semi-PLSA

By performing PLSA only on a subset of documents as described in chapter 3 obviously lead to faster results. However, propagating the results of sampled documents to the other documents incurs a non-negligible computation cost. The overall speed-up achieved is shown in Fig. 4.4. As expected, the smaller the sampled size, the faster the computation. This result includes the cost of computing the missing values through the propagation mechanism.

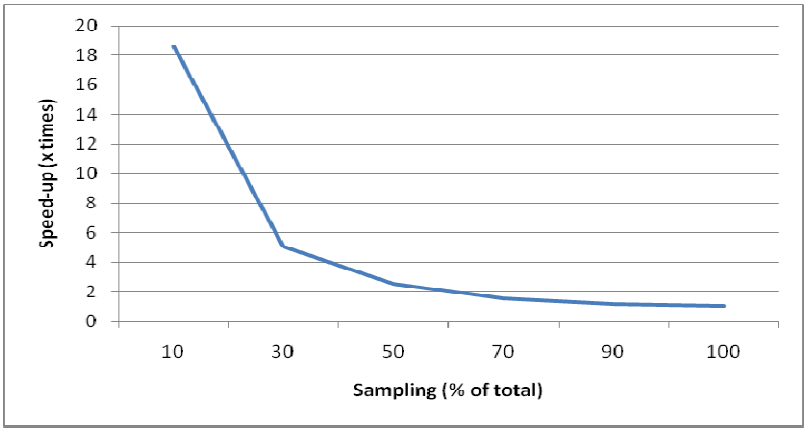


Fig 4.4: Average speed-up obtained for different sampling size (as percentage of original data).

To assess the loss of accuracy, the problem is to compare two probability distributions. The metric used in Fig 4.5 is the average absolute distance between the probabilities obtained with the original PLSA and the sampling-based PLSA. That is, for every document d, the original topic coverage $p(z|d)_{orig}$ is compared to the coverage $p(z|d)_{approx}$ obtained with the approximate algorithm. As expected, the smaller the sampled size, the less accurate the results are. However, since all documents will be bucketed, then the system allows some loss in accuracy, as long as the average loss is smaller than the bucket size.

For instance, if each topic is broken into 10 buckets of equal size, then every document with topic coverage (for a specific topic T) between 0 and 0.1, will be assigned to the first bucket. Similarly, every document with topic coverage between 0.1 and 0.2 will be assigned to the second bucket, and so on. That means, depending on the size of the bucket (0.1 here), it does not matter whether the true topic coverage for a specific document is 0.05 or 0.06, because the document will still be assigned to the same bucket.

The general rule is that an acceptable accuracy loss must be smaller than the bucket size.

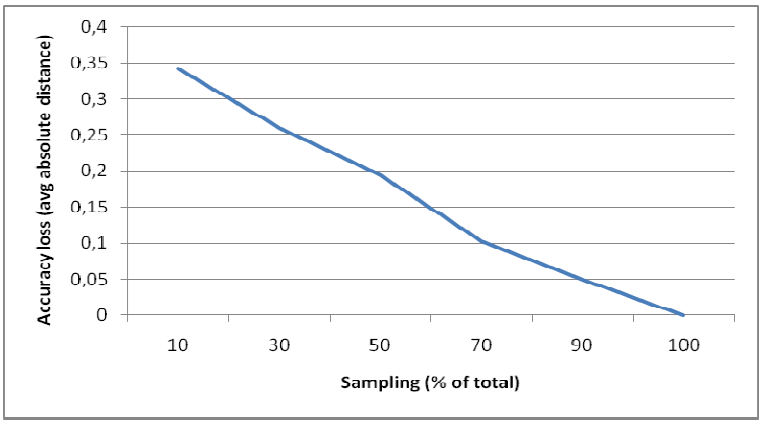


Fig 4.5: Accuracy loss obtained for different sampling size (as percentage of original data).

4.2. TYPICAL APPLICATIONS

To demonstrate the usefulness of our system, we present two typical scenarios in which the possibility of adding ad hoc dimensions and highlighting values are two efficient tools. Generally speaking, the usefulness of our system, compared to a typical spreadsheet system, is mainly the integrated support for text data. Besides, all the typical applications of a spreadsheet system could be implemented on our system.

4.2.1. Extending regular dimension set

In the context of laptop reviews, some characteristics such as 'popularity', or 'slickness' may not be present in the structured data. While some reviews try to integrate these aspects in the form of rating (e.g. five stars for a good laptop), these ratings may not be consistent across reviewers, and may not correspond to what the user has in mind. By defining its own topics, the user directs the system to assign every record with additional ratings.

While not as sophisticated as other state-of-the art text mining system, our system integrates seamlessly such facility within the global framework of OLAP mining. Fig 4.6 shows how to extend a set of laptop reviews, containing only 'screen size', 'brand', 'battery life' and 'price' information, with additional dimensions such as 'popularity' and 'portability'.

4.2.2. Comparing entities

The most profitable use of the system is to compare different entities. Specifically, when additional topic dimensions are defined, the DDA extension of the system provides an easy way to compare different entities. Indeed, by suggesting 'interesting' dimensions to distinguish these entities, and by highlighting the 'surprising' cells, the system is able to quickly provide an accurate comparative description of different entities. Additionally, the topic comparison feature generates a summary of this comparison by presenting the average topic values for each entity.

If we consider the task of aviation safety analysts, it is a common task for them to analyze thousands of flights, each having a set of structured records and free form texts. In this context, analysts would choose several types of flight anomalies (e.g. 'equipment problem', 'weather') as topics. Then, our system can quickly produce a comparison of these flights at a topic level ("for each airport, what is the most common flight anomaly?"). Analysts can further explore each topic to discover how important is the flight anomaly ("for each airport, how severe is the topic 'equipment problem'?).

The system does not substitute for the analyst's expertise, but rather complements it, by providing features which quickly bring the analyst to the interesting part of her data. Fig 4.7 shows an aggregated view of airport for different flight anomalies.

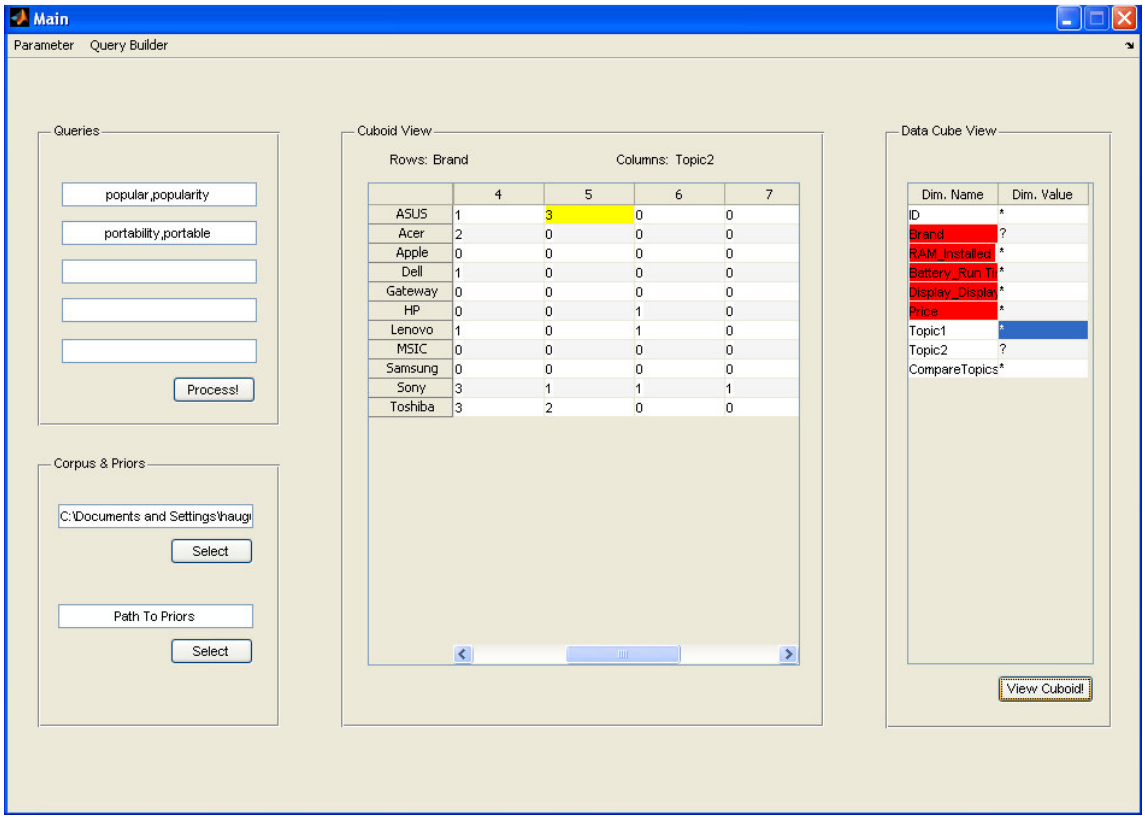


Fig 4.6: Laptop reviews expanded on brand and portability (Topic 2).

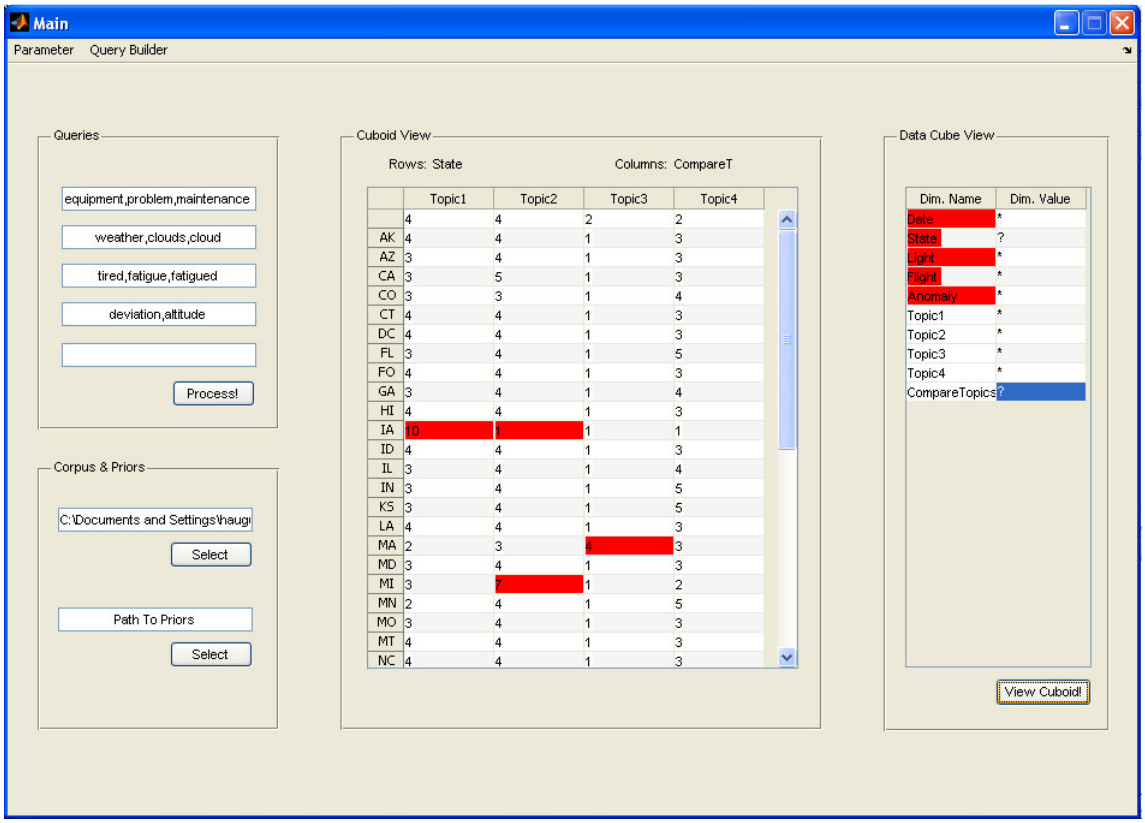


Fig4.7: Comparing airport for different flight anomalies

CHAPTER 5: RELATED WORKS

To a large extent, this work is related to the question of how to extract information from text data. That is, structuring text data is essentially a process of finding relevant information within text. We presented several methods based on the PLSA algorithm to extract data from text, but other methods exist.

For instance, entity recognition is the process of classifying atomic text elements into predefined categories (e.g. person name). This technique transforms a free form text into a set of entities, which can be processed later by the system. Several authors ([6],[13],[10], [11]) have pioneered the standard architecture for an entity extraction system. The major components of such system are the pattern extractor, which extracts fragments of sentence matching a predefined pattern, and the feature generator, which filters out irrelevant fragments. Pantel and Pennacchiotti, in [12], proposed a state-of-the-art entity recognition system, which relies on ensemble voting and multiple pattern extractors.

Faceted search is a more advanced set of techniques which enables text documents to be filtered by some attributes (e.g. 'authors' for books). As such, faceted search enables the conversion of free form text documents to structured documents. Several milestone projects ([3], [1], [17], [2]) have contributed to the progress of faceted search in general. In particular, Xu & al, in [8], proposed a system to generate summaries of documents, using a faceted search interface. Their system relies on user input to create facets (i.e. topics) and their topic modeling based algorithm to create the content of each facet.

Our system complements these previous works by integrating the structured documents into a larger task, namely the discovery driven analysis.

On the other hand, several works related to data analysis are relevant to our system. Indeed, most of the analysis performed by the system relied on previous work in OLAP and discovery driven analysis.

As explained in the introductory chapter, OLAP is a set of techniques which enables viewing multidimensional data at different levels of granularity. Most of the work on OLAP has been done in industry, resulting on several widely use products. At the core of OLAP technology is the concept of multidimensional databases ([4]). Multidimensional databases represent data as numeric facts, which are categorized by dimensions. Each fact is indexed by its dimension values, and multiple dimension values can be aggregated into a single group, thus creating a new numeric fact. This aggregation operation is performed according to a hierarchy built on the dimension.

While most of the work in the multidimensional database field is concerned with performance issues ([5]), several works have been published in the Data Mining area on how to exploit OLAP technology to support deeper analysis. Sarawagi & al, in [16], proposed discovery driven analysis and further extend it in [14] and [15]. Their system estimates analyst's expectations about the data by applying a maximum-entropy model. Their system also provides several features to explain difference between values in the data cube, based their user model.

Again, our system complements these works by extending them to new type of unstructured data, such as text.

CHAPTER 6: CONCLUSION

In this work, we presented our solution for extending discovery driven analysis on text data, and provided an implementation of this solution. Specifically, we introduced our method to process free form text into structured data, and our technique for improving the navigation within these structured data. We discussed the computational challenges and how they impacted the implementation of our solution. Finally, we tested the scalability of our system and provided some typical applications.

The main contribution of this work is the idea of allowing analysts to explore their text data, in an ad hoc fashion. Specifically, our system enables rigorous OLAP analysis on text data, while maintaining enough flexibility for analysts to input their knowledge about the data.
As an additional contribution, this thesis introduces the technical challenges involved when performing discovery driven analysis on text data, and provides a fully functional system as a solution for these challenges.

Future work will feature new models to better structure text data. For instance, one may be interested in a model which treats each topic independently. Such model would allow incremental computation of queries, instead of computing everything from scratch, as semi-PLSA does. The authors are currently experimenting with the Kullback-Leibler divergence as a new model.

Another direction for future work is to adapt more complex 'interestingness models'. Indeed, the current model treats each column independently, and does not take into account the neighbor cuboids in the computation of DDA indicators. Overcoming such limitations, while maintaining good computation performance, would allow more accurate and more robust modeling of 'interestingness'.

A final direction is the challenge of scaling to very large datasets. The methods presented in this work managed the size of data by reducing the accuracy on the results. However, these methods maintain a reasonable accuracy only for datasets with a thousand documents. Parallelizing the computations is a promising idea to handle larger datasets.

REFERENCES

[1] Robert G. Capra and Gary Marchionini, "The relation browser tool for faceted exploratory search," , 2008, pp. 420-420.

[2] Debabrata Dash, Jun Rao, Nimrod Megiddo, Anastasia Ailamaki, and Guy Lohman, "Dynamic faceted search for discovery-driven analysis," , 2008, pp. 3-12.

[3] Ame Elliott, "Flamenco image browser: using metadata to improve image search during architectural design," , 2001, pp. 69-70.

[4] Jim Gray et al., "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals," *Data Min. Knowl. Discov.*, vol. 1, no. 1, pp. 29-53, 1997.

[5] Jiawei Han, *Data Mining: Concepts and Techniques.*: Morgan Kaufmann Publishers Inc., 2005.

[6] Marti A. Hearst, "Automatic acquisition of hyponyms from large text corpora," , 1992, pp. 539-545.

[7] Thomas Hofmann, "Probabilistic latent semantic indexing," , 1999, pp. 50-57.

[8] Xu Ling, Qiaozhu Mei, ChengXiang Zhai, and Bruce Schatz, "Mining multi-faceted overviews of arbitrary topics in a text collection," , 2008, pp. 497-505.

[9] Yue Lu and Chengxiang Zhai, "Opinion integration through semi-supervised topic modeling," , 2008, pp. 121-130.

[10] Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain, "Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge," , 2006, pp. 1400-1405.

[11] Marco Pennacchiotti and Patrick Pantel, "A Bootstrapping Algorithm for automatically harvesting semantic relations," , 2006, pp. 87-96.

[12] Marco Pennacchiotti and Patrick Pantel, "Entity extraction via ensemble semantics," , 2009, pp. 238-247.

[13] Ellen Riloff and Rosie Jones, "Learning dictionaries for information extraction by multi-level bootstrapping," , 1999, pp. 474-479.

[14] Sunita Sarawagi, "Explaining Differences in Multidimensional Aggregates," , 1999, pp. 42-53.

[15] Sunita Sarawagi, "User-cognizant multidimensional analysis," *The VLDB Journal*, vol. 10, no. 2-3, pp. 224-239, 2001.

[16] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo, "Discovery-Driven Exploration of OLAP Data Cubes," , 1998, pp. 168-182.

[17] M.C. Schraefel, Max Wilson, Alistair Russell, and Daniel A. Smith, "mSpace: improving information access to multimedia domains with multimodal exploratory search," *Commun. ACM*, vol. 49, no. 4, pp. 47-49, 2006.