TEXTDIVE: CONSTRUCTION, SUMMARIZATION AND EXPLORATION OF MULTI-DIMENSIONAL TEXT CORPORA

BY

QI WANG

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, 2016

Urbana, Illinois

Adviser:

Professor Jiawei Han

ABSTRACT

With massive datasets accumulating in text repositories (*e.g.*, news articles, customer reviews, *etc.*), it is highly desirable to systematically utilize and explore them by data mining, NLP and database techniques. In our view, documents in text corpora contain informative explicit meta-attributes (*e.g.*, category, date, author, *etc.*) and implicit attributes (*e.g.*, sentiment), forming one or a set of highly-structured multi-dimensional spaces. Much knowledge can be derived if we develop effective and efficient multi-dimensional summarization, exploration and analysis technologies.

In this demo, we propose an end-to-end, real-time analytical platform TextDive for processing massive text data, and provide valuable insights to general data consumers. First, we develop a set of information extraction, entity typing and text mining methods to extract consolidated dimensions and automatically construct multi-dimensional textual spaces (*i.e.*, *text cubes*). Furthermore, we develop a set of OLAP-like text summarization, data exploration and text analysis mechanisms that understand semantics of text corpora in multi-dimensional spaces. We also develop an efficient computational solution that involves materializing selective statistics to guarantee the interactive and real-time nature of TextDive.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I would like to give special thanks to my advisor, Dr. Jiawei Han, for his continuous support on my research and inspiring advice on my topic. This is a joint work with Fangbo Tao, and several other researchers. The TextDive part is led by Fangbo Tao and me, coauthered with Honglei Zhuang, Chi Wang, Taylor Cassidy, Lance Kaplan, Clare Voss and Jiawei Han; the CASeOLAP part is led by Fangbo Tao, coauthored with me, Chi Wang, Honglei Zhuang, Taylor Cassidy, Lance Kaplan, Clare Voss, and Jiawei Han. I would also like to thank other colleagues, Jialu Liu and Jingbo Shang for their helpful advice on this work.

Research was sponsored in part by the U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), National Science Foundation IIS-1017362, IIS-1320617, and IIS-1354329, HDTRA1-10-1-0120, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies of the U.S. Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 RELATED WORK 4

CHAPTER 3 PRELIMINARIES 5
3.1 Text Cube Basics 5
3.2 Problem Definition 7

CHAPTER 4 MAJOR MODULES 9
4.1 Data Preprocessing 9
4.1.1 Typed Entity Extraction 10
4.1.2 Mining Phrases as Representation 10
4.1.3 Entity Extraction Using NLP 10
4.1.4 Entity Extraction Using Hierarchical Topic Ontology . . . 11
4.2 Functional Modules 11
4.2.1 Context-aware Semantic Summarization 12
4.2.2 Mining Outlier Document 13
4.2.3 Intelligent Exploration 14
4.2.4 Real-time Computational Engine 15

CHAPTER 5 SYSTEM DEMONSTRATION 16
5.1 System Design 16
5.2 Use Case: Business Consulting 17

CHAPTER 6 COMPUTATIONAL METHODOLOGY 19
6.1 Overview 19
6.2 Hybrid Offline Materialization 20
6.2.1 Cost Estimation 21
6.2.2 Simple Greedy Algorithm 23
6.2.3 Utility-Guided Greedy Algorithm 24
6.3 Optimized Online Processing 25

CHAPTER 7 CONCLUSION 27

REFERENCES 28

CHAPTER 1

INTRODUCTION

In the big data era, massive amount of text data has been created rapidly from both public domains (*e.g.*, news articles, customer reviews, social media posts *etc.*) and private organizations (*e.g.*, business reports, internal logs, *etc.*). It quickly becomes unmanageable for humans to understand tens of millions of documents. To systematically analyze such large amount of textual data, it is often favorable to manage the massive text corpora (and its metadata) in multi-dimensional semantic spaces (*i.e.*, *text cubes* [1]), where the dimensions correspond to 1) a set of meta-attributes (*e.g.*, category, date, time) and/or 2) extracted implicit information (*e.g.*, sentiment, topic) associated with each document. Similar to traditional numerical data cube techniques, some recent studies ([1, 2, 3]) have developed numerical measures (*e.g.*, count, probability, *etc.*) associated with a set of keywords in the documents. However, these keyword-based measures neglect the semantics embedded in the rich text and therefore are not very insightful. To further harvest knowledge in massive text corpora, a more sophisticated analytical engine needs to be built to summarize, explore and mine the multi-dimensional textual spaces.

In our recent research, we have made progress towards the construction of multi-dimensional textual spaces. We have studied how to extract entities from closed domains [4] and assign proper types to them [5], which forms the base of constructing multiple dimensions. Moreover, an effective phrase mining approach, *SegPhrase* [6], is incorporated into the construction and performs as part of the representation of text. Combined with existing meta-attributes, a multi-dimensional textual space is constructed and stored with efficient indexing prepared to guarantee the low latency.

To effectively facilitate the constructed multi-dimensional textual spaces, we also explore three disjoint OLAP-like operations on text data. First, we have studied the automatic text summarization problem in multi-dimensional spaces and proposed CASeOLAP [7] (Context-aware Semantic OLAP) to provide informative summarization given a user query. The summarization can be either

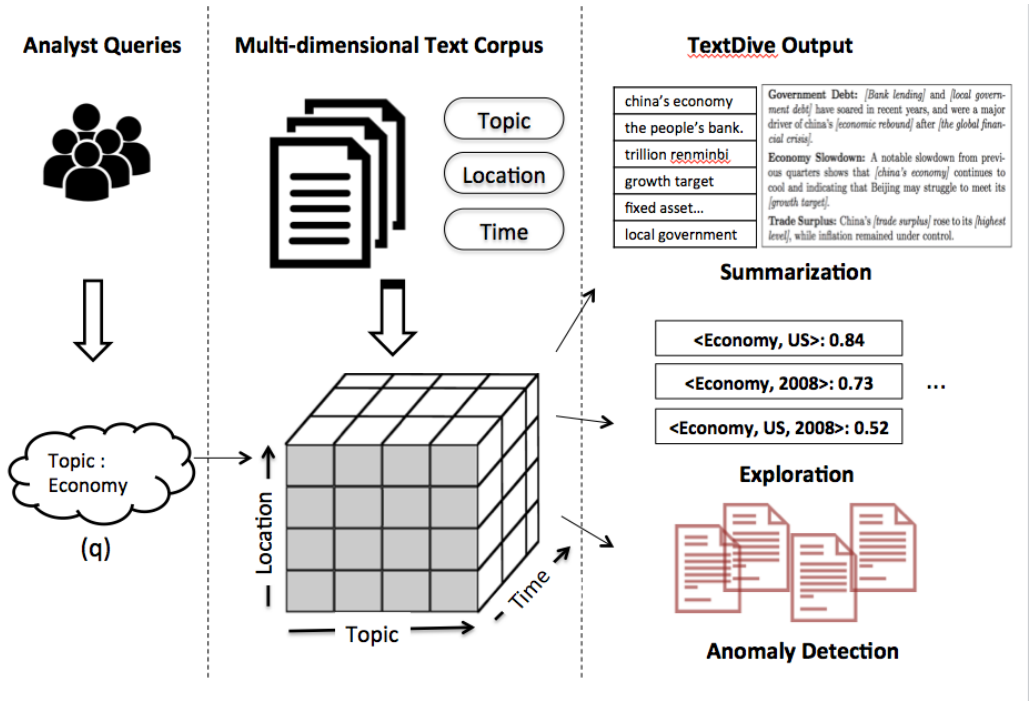

Figure 1.1: TextDive **illustration**

top-k representative phrases or extracted sentences covering various aspects of documents within the query (called a *cell*). The second operation we develop is anomaly detection in multi-dimensional textual space. It helps to find anomalous cells/documents in the space which are likely to provide information and insights overlooked by a general summarization. Moreover, due to the large number of query possibilities, users may have difficulties finding interesting subsets of the corpora to dive into, we develop intelligent exploration mechanism to guide users' exploration for insightful result. Two interestingness measures, *explanation score* and *in-cell diversity*, are proposed to achieve proper guidance.

System Illustration: The TextDive system is designed with the illustration shown in Figure 1.1. Suppose a multi-dimensional text corpora is constructed from *New York Times* news articles with three hierarchical dimensions: *Location*, *Topic* and *Time*. An analyst may pose a multi-dimensional query (q): ⟨Economy⟩, corresponding to dimension *Topic*. The summarization module generates top representative phrases and sentences to concisely describe the gist of documents in the cell. The anomaly detection module marks the outlier documents discussing the collapse of Greek economy to indicate those documents are semantically anomalous. Then the exploration module outputs top sub-cells, ⟨Economy, US⟩ and ⟨Economy, 2008⟩, that credited most for the summarization of cell

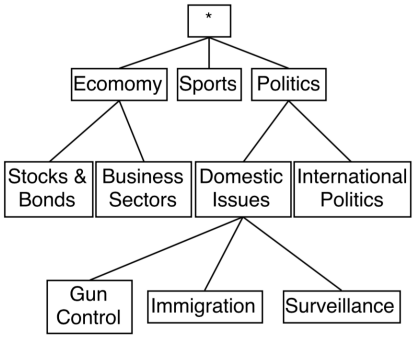

Figure 1.2: Hierarchy of *Topic*

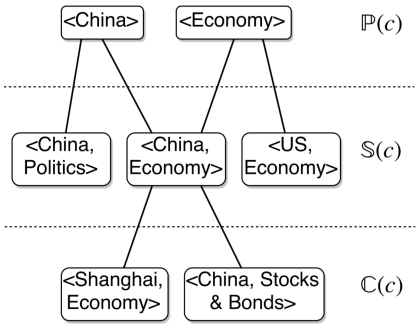

Figure 1.3: *Context* of cell ⟨China, Economy⟩

⟨Economy⟩.

In this demo, we show how a quality multi-dimensional textual space can be constructed and how various OLAP-like operations can be effectively performed on TextDive. The system is based on our recent research and the development of several previous systems: EventCube [8] and NewsNetExplorer [9].

CHAPTER 2

RELATED WORK

Several important pieces of related work [2, 10, 6] have been introduced in Chapter 1. In this chapter we discuss other previous work related to TextDive, mainly about mining representative information from text corpora. Text Cube [1] takes a multi-dimensional view of textual collections and proposed OLAP-style *tf* and *idf* measures. Besides that, [11, 12] also proposed OLAP-style measures on term level using only local frequency, which cannot serve as effective semantic representations. [13, 14] focused on interactive exploration framework in text cubes given keyword queries, without considering the semantics in raw text. Several multi-dimensional analytical platforms [15, 8] are also constructed to support end-to-end textual analytics. However, the supported measures are numerical term-level ones. Another related topic is Faceted Search [16, 17, 18, 19], which dynamically aggregates information for an ad-hoc set of documents. the aggregation is usually conducted on meta data (called *facets*), not document content.

Quality phrase mining is also extensively studied by NLP community [20, 21] and Data Mining community [22, 23]. They either utilize sophisticated NLP features or use various statistical measures to estimate phrase quality. Those phrase mining methods serve as the candidate generation step for our framework.

In order to apply CASeOLAP, a fundamental step is to construct dimensions for textual data, several NLP-based methods like Named-entity Set Expansion [4, 24] and Information Extraction [25, 26, 27] can be leveraged to generate values framework dimensions in Text Cubes.

CHAPTER 3

PRELIMINARIES

In this chapter, we formally define the concept of text cube, the CASeOLAP problem, the representative phrase mining task, and the three ranking criteria.

3.1 Text Cube Basics

Similar to traditional multi-dimensional data cubes, a *text cube* [1] is a data model but over text collection $\mathcal{DOC}$ that has metadata for documents. The metadata can be either extrinsic attributes of the documents, such as classification taxonomy, or intrinsic information extracted from the documents, such as named entities mentioned in them. In this work, we focus on single-valued categorical metadata, and leave other types of metadata to future work. We assume there are n categorical attributes (*i.e.*, *dimensions*) associated with each document in $\mathcal{DOC}$. For example, a news article in *NYT* corpus is represented as (*Jan 2012*, *China*, *Economy*, *'After a sharp economic slowdown through much of last year...'*). It denotes that the *'Time'* of the article is *Jan 2012*, *'Location'* is *China* and *'Topic'* is *Economy*.

The dimensions provide valuable context for each document. Like a traditional data cube, all distinct values of one dimension are organized in a *dimension hierarchy*. For i-th dimension, the dimension hierarchy $\mathcal{A}_i$ is a tree where the root is denoted as '$*$'. Each non-root node is a value in that dimension. The parent node of a dimension value a_i is denoted as $par(a_i)$, and the set of direct descendants of a_i is denoted as $des(a_i)$. For example, Figure 1.2 illustrates a partial dimension hierarchy about 'topics' in *NYT* corpus. It is a tree of height 4, with a root node '$*$'. $par($*Gun Control*$)$ = *Domestic Issues* and $des($'$*$'$) = \{$*Economy*, *Sports*, *Politics*$\}$.

Formally, we have the following definition.

Definition 1 (Multi-dimensional Text Cube) *A text cube is defined as* $\mathcal{TC} = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n, \mathcal{DOC})$*, where* $\mathcal{A}_i$ *is a dimension*

hierarchy. Each document is in the form of $(a_1, a_2, \ldots, a_n, d)$*, where* $a_i \in \mathcal{A}_i \backslash \{*\}$ *is a dimension value for* $\mathcal{A}_i$ *and* d *is a string of the content. A cell* c *in the cube is represented as* $(a_1, \ldots, a_n, \mathcal{D}_c)$*, where* $a_i \in \mathcal{A}_i$*, and* $\mathcal{D}_c \subseteq \mathcal{DOC}$ *is the subset of documents contained in cell* c*. For notation simplicity, we use* $\langle a_{t_1}, \ldots, a_{t_k} \rangle$ *to refer to a cell with non-*$*$ *dimension values* $\{a_{t_1}, \ldots, a_{t_k}\}$*.*

Example 3. Fig. 3.1 illustrates a mini example of news article text cube, with 3 dimensions (Time, Location and Topic) and 9 documents d_1–d_9. The Time dimension is derived from extrinsic attribute but Location and Topic are extracted by information extraction as in [9]. We list 7 non-empty cells, where the top four are leaf cells without '$*$' dimensions, *e.g.*, (Jan 2012, China, Economy, $\{d_1, d_2\}$). The root cell (entire corpus) is represented as ($*$, $*$, $*$, $\{d_1$–$d_9\}$). ■

Dimensions			**Text Data**
Time	**Location**	**Topic**	$\mathcal{DOC}$
Jan 2012	China	Economy	$\{d_1, d_2\}$
Aug 2012	China	Economy	$\{d_3, d_4, d_5\}$
Aug 2012	US	Gun Control	$\{d_6, d_7\}$
Nov 2012	US	Economy	$\{d_8, d_9\}$
$*$	China	Economy	$\{d_1, \ldots, d_5\}$
Aug 2012	$*$	$*$	$\{d_3, \ldots, d_7\}$
$*$	$*$	$*$	$\{d_1, \ldots, d_9\}$

Figure 3.1: Mini Example of *NYT* Corpus

Text cube provides a framework for organizing text documents using meta-information. In particular, the cell space defined above embeds the inter-connection between different subsets of text. To capture those semantically close cells, we define *context* of a cell c as a composition of three parts.

Definition 2 (Cell Context) *The context of cell* $c = \langle a_{t_1}, \ldots, a_{t_k} \rangle$ *is defined as* $\mathbb{P}(c) \bigcup \mathbb{S}(c) \bigcup \mathbb{C}(c)$*, where:*

- *Parent set is defined as* $\mathbb{P}(c) = \{\langle a_{t_1}, \ldots, par(a_i), \ldots, a_{t_k} \rangle | i \in t_1, \ldots, t_k\}$*. Each parent cell is found by changing exactly one non-*$*$ *dimension value in cell* c *into its parent value;*
- *Children set is defined as* $\mathbb{C}(c) = \{c' | c \in \mathbb{P}(c')\}$*. Each child cell is found by either changings one* $*$ *value into non-*$*$ *or by replacing it by one of the child values; and*

- *Sibling set is defined as* $\mathbb{S}(c) = \{c' | \mathbb{P}(c) \bigcap \mathbb{P}(c') \neq \emptyset\}$. *Each sibling cell must share one parent with cell* c.

Example 4. Fig. 1.3 illustrates the partial context of cell $c =$⟨China, Economy⟩. The parent set $\mathbb{P}(c)$ contains ⟨China⟩ and ⟨Economy⟩, sibling set $\mathbb{S}(c)$ has ⟨China, Politics⟩ and ⟨US, Economy⟩ and children $\mathbb{C}(c)$ contains ⟨Shanghai, Economy⟩ and ⟨China, Stocks & Bonds⟩. ■

3.2 Problem Definition

The core module of the demo, CASeOLAP, deals with the problem of online analytical processing with representative phrases, in particular within multi-dimensional text cube. A phrase is a multi-word sequence served as an integral semantic unit. The representative phrases for a cell, are the phrases that characterize the semantics of the selected documents. There is no universally accepted standard of being *representative*. Here we operationalize a definition in terms of three criteria.

- **Integrity:** An integral phrase must satisfy two conditions: (i) the multiple words in a phrase collocate together much more frequently than expected from random chance, and (ii) the phrase is a complete semantic unit, rather than a subsequence of another equally-frequent phrase.
- **Popularity:** A phrase is popular if it has a large number of occurrences. Representative phrases for a cell, in particular, should appear with some frequency within the documents of that cell. Very low frequency phrases within a cell do not contribute substantially to its semantics and so are not considered representative.
- **Distinctiveness:** High-popularity phrases that appear in many different cells constitute background noise, *e.g.*, 'earlier this month' and 'focus on'. Representative phrases should distinguish the target cell from its context, therefore provide more salient information to help users filter the noise. Distinctiveness is particularly critical in CASeOLAP, since analysts often navigate through the whole collection to find subsets of interest. Non-distinctive phrases will appear in many cells and offer redundant information.

However, none of the previous work has followed all three criteria. MCX [2, 10] follows *distinctiveness* (in a rough sense) and ignores *popularity* and *integrity*.

SegPhrase [23] addresses *integrity* in global quality phrase mining, but the notion of *popularity* and *distinctiveness* with respect to a target cell is not applicable to that problem setting. This paper proposes a new measure to evaluate all three criteria.

Within the whole ranked phrase list, top-k representative phrases normally have higher value for users in text analytics. As a further matter, the top-k query also enjoys computational superiority, so that users can conduct fast analytics. For these reasons, we define the problem as follows.

Definition 3 (CASeOLAP in Text Cube) *Given a multi-dimensional text cube* $\mathcal{TC} = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n, \mathcal{DOC})$*, CASeOLAP takes* $c = (a_1, \ldots, a_n, \mathcal{D}_c)$ *as a query, and outputs top-*k *representative phrases based on the integrity, popularity and distinctiveness criteria.*

CHAPTER 4

MAJOR MODULES

In this chapter, we will introduce the major modules of TextDive including the preprocessing module, which generates the structured corpora of New York Times news corpora, and the major functional modules of the TextDive.

4.1 Data Preprocessing

We combine the explicit attributes and extracted typed entities as dimensions. For example, in the *New York Times* dataset, five dimensions are created including *Location*, *Time* and *Topic* from annotated meta-data and *Person* and *Organization* from extracted typed entities. All dimensions are designed to be hierarchical to support OLAP-like drill-down/roll-up queries. An example hierarchy of dimension *Topic* is shown in Figure 1.2, where $*$ means 'all' values for the dimension. After dimensions are determined by users, the system automatically associates each document with one or multiple values in each dimension. Therefore, given a multi-dimensional query, a set of documents will be retrieved for further analysis. Corresponding indexing is created to facilitate quick access of documents with multi-dimensional queries.

The constructed multi-dimensional space provides a framework for organizing documents using dimensions. In particular, the cell space defined implies the inner-connection between subsets of documents, *i.e.*, cells. We define *context* to denote such connections. The connections are built if two cells share the same dimension value or have direct link in one of their hierarchical dimension. For example, the *context* of cell ⟨Economy, China⟩ is illustrated in Figure 1.3, in which ⟨Economy⟩ is a parent cell, ⟨Economy, US⟩ is a sibling cell and ⟨Economy, Shanghai⟩ is a child cell.

4.1.1 Typed Entity Extraction

For text datasets without sufficient meta-attributes, we develop several methods to extract entities as dimension values. For *Location*, *Person* and *Organization*, we simply use Stanford NER tool[1]. For other customized types, due to the potential sparsity of the mentions of that type, we apply the *Semantic Pattern Graph* method [4] to tackle mention sparsity and extract entities by giving a small set of seed entities. The method leverages web signals to enhance the entity coverage and precision. ClusType [5] is later applied to assign types of extracted entities, which clusters surrounding lexical patterns together to enhance typing precision. After extracting typed entities, we use freebase ontology[2] to build hierarchical relationships between entities of the same type. The resulting inter-linked entities can be used as dimensions in our multi-dimensional framework.

4.1.2 Mining Phrases as Representation

Quality phrases often have better semantic meaning than unigrams. As part of construction, we associate a list of phrases with each document using *SegPhrase* [6]. In a nutshell, SegPhrase first generates frequent phrase candidates according to global popularity requirements, and then estimates phrase quality based on multiple statistical features. The mined phrases are used widely in summarization and exploration operations.

4.1.3 Entity Extraction Using NLP

From the free text or the text segments of the textual attributes in the integrated structured and text news data, natural language processing (**NLP**) (especially Information Extraction) tools are used to extract essential entities such as time, location, person, organization. Moreover, concept hierarchies (*i.e.*, higher-level entities) are associated with extracted entities (*e.g.*, Chicago is associated with state: Illinois) based on a user- or expert-provided dictionary.

[1] http://nlp.stanford.edu/software/CRF-NER.shtml
[2] http://www.freebase.com/

Table 4.1: Top-10 representative phrases for two user queries

⟨**US, Gun Control**⟩	⟨**US, Immigration**⟩
gun laws	immigration debate
the national rifle association	border security
gun rights	guest worker program
background check	immigration legislation
gun owners	undocumented immigrants
assault weapons ban	overhaul of the nation's immigration laws
mass shootings	legal status
high capacity magazines	path to citizenship
gun legislation	immigration status
gun control advocates	immigration reform

4.1.4 Entity Extraction Using Hierarchical Topic Ontology

For *Topic* extraction of NewsNetExplorer we have developed a method that construct a topical hierarchy from a collection of text. The framework called **CATHY** (Construct A Topical HierarchY) is a recursive clustering and ranking approach for topical hierarchy generation. In the news collection, many pieces can describe the same topic; meanwhile, different topics may reflect the same fact at different levels of granularity. The aim of this method is to construct a hierarchy where each topic is represented by a ranked list of phrases, such that a child topic is a subset of its parent topic. This strategy also works well specifically for news data since news articles tend to use different phrases to report the same topics.

4.2 Functional Modules

After constructing the multi-dimensional textual space, TextDive is designed and implemented to support deeper analysis other than faceted search. Similar to traditional *data cube* and *OLAP* techniques, multiple text-based measures are supported and implemented to guarantee real-time analysis. In this section, we introduce multi-dimensional automatic text summarization in Section 4.2.1, anomaly detection in Section 4.2.2 and intelligent exploration in Section 4.2.3 .

Government Debt: *[Bank lending]* and *[local government debt]* have soared in recent years, and were a major driver of china's *[economic rebound]* after *[the global financial crisis]*.

Economy Slowdown: A notable slowdown from previous quarters shows that *[china's economy]* continues to cool and indicating that Beijing may struggle to meet its *[growth target]*.

Trade Surplus: China's *[trade surplus]* rose to its *[highest level]*, while inflation remained under control.

Figure 4.1: Top-3 key sentences for ⟨Economy, China⟩

4.2.1 Context-aware Semantic Summarization

When a political analyst tries to understand strategies of different candidates in the *US Presidential Election* using NYT articles, the very first step is likely to be summarizing the related news for each candidates, e.g., *Trump* and *Clinton*. Finding key concepts and key sentences largely improves his/her productivity on analytical such tasks. Therefore, we propose Context-aware Semantic OLAP (CASeOLAP) to provide *top-k representative phrases* and *sentences* for summarization. Since the neighboring cells (e.g., different presidential candidates) in the multi-dimensional space often share similar background topics, it is desirable to generate summarizations that characterize the set of documents in the cells and be distinguished from those of other cells (*i.e.*, context) in the cube. The phrase summarization of two sample cells are shown in Table 4.1, the sentence summarization of cell ⟨Economy, China⟩ is shown in Figure 4.1.

To generate top-k representative phrases, we consider three major factors, *integrity*, *popularity* and *distinctiveness*, and using the geometric mean as the ranking criteria.

$$r(p, c) = \sqrt[3]{int(p, c) \cdot pop(p, c) \cdot dis(p, c)} \tag{4.1}$$

The distinctiveness score considers the comparison of phrase p in cell c and its sibling cells. It effectively removes the background phrases and make the summarization very representative.

Furthermore, we use sentences for automatic text summarization using the same three criteria. In a nutshell, our algorithm first mines top phrases and uses them to construct weighted edges between the sentences mentioned in the cell documents. An affinity matrix for all sentences then can be created from the cell. The representative phrases make the semantically close sentences also close in the network. Then we apply *spectral clustering* to find k clusters and select the central

Table 4.2: Examples of outlier news articles published in "Health" section of New York Times.

Rank	Outlier document snippet
1	*CHICAGO (AP) States with the most gun control laws have the fewest gun-related deaths, according to a study that suggests sheer quantity of measures might make a difference ...*
2	*ATLANTA There's more evidence that U.S. births may be leveling off after years of decline. The number of babies born last year only slipped a little, and preliminary government figures ...*
...	...

sentence into the summarization. The extracted sentences are both representative and mutually exclusive semantically. In Figure 4.1, the representative phrases are marked by square brackets. We notice that sentences with presence of representative phrases tend to carry richer semantics of the target cell.

4.2.2 Mining Outlier Document

In addition to summarization of documents, users may also be interested in anomaly documents, namely documents with significant deviated topic from the majority of the given set of documents. For example, the same political analyst is interested in the articles related to *Trump* with special topics or extreme political opinions, to have a hollistic picture of *Trump*'s strategy. Mining such outliers may complement document summarization, as outliers unveil potential inconsistencies among documents, implying possible errors, unexpected information, or novel insights.

Given a user query (i.e., cell), we output the a ranked list of possible document outliers topically deviating from the queried corpus. By examining the top-ranked documents, the user may discover interesting insights or potential flaws. As an example, Table 4.2 shows a ranked list of outlier documents from a user query of ⟨Health⟩ cell in NYT news articles. The article ranks top in this example is one about gun control policy and its correlation to gun-related deaths. This document is very different from the other documents in the cell, which are more relevant to topics such as medical studies on cancers, or health insurance. Therefore, the topic it covers is very unlikely to be represented in a summarization method but

may still provide deeper insights, that mental health plays a role in the correlation between gun-related deaths and gun control policy.

To find out the anomaly documents, we employ word embedding technique to convert the corpus into a bag of normalized embedded vectors. Then we identify semantic focuses of the corpus and define the probability that a word belongs to the semantic focuses as $P(\varphi_{ij} = 1|w_{ij})$, where w_{ij} is the j-th word in the i-th document. We model the total number of words belonging to the semantic focuses as a random variable n_i^{φ} drawn from a Poisson-Binomial distribution, and define its lower limit with high confidence θ as:

$$q_\theta(n_i^{\varphi}) = \max_q P(n_i^{\varphi} \geq q) \geq \theta$$

Accordingly we propose a document outlierness measure:

$$\Omega_{\theta\text{-q}}(d_i) = 1 - \frac{q_\theta(n_i^{\varphi}) + 1}{|d_i| + 1}$$

which favors documents without words that belong to the semantic focus of the corpus.

4.2.3 Intelligent Exploration

Given a constructed multi-dimensional text corpora, there can be millions of cells (exponential to the number of dimensions) that users can query. Thus, it becomes challenging for users to find interesting cells. After viewing the summarization of ⟨Healthcare⟩ documents, the political analyst may want to know which presidential candidate contributes most to the heathcare discussion. Or he may want to know, which presidential candidate, has the most self-contradictory opinions on healthcare issues.

We denote the first type of exploration as *explanations finding*. Multiple drill-down paths can be performed and TextDive examines all potential subcells (e.g., different presidential candidates), evaluates the contribution by *intervention test*. The intervention test removes the explanation from the original query (e.g., ⟨Healthcare⟩) and evaluates the semantic diversion degree of the remaining cell. The subcells that divert the original cell most are the top explanations. As a real example, the top-4 *explanations* of ⟨Economy⟩ is shown in Table 4.3, in which ⟨Economy, US⟩ is drilled down through Location and ⟨Economy, 2008⟩

is drilled down through Time.

Explanation	Score
⟨Economy, US⟩	0.84
⟨Economy, 2008⟩	0.73
⟨Economy, 2008, US⟩	0.52
⟨Economy, China⟩	0.33

Table 4.3: Top explanations of ⟨Economy⟩ with scores

The second type of exploration is denoted as *in-cell diversity*. A cell with higher internal semantic discrepancy is regarded as more "interesting" than one with similar semantics between its subcells. Therefore, *in-cell diversity* measures the internal discrepancy of a set of cells and recommend the interesting ones to a user. Since multiple drill-down paths can be applied to the same cell, we evaluate the diversity of these paths and recommend drill-down paths to users.

4.2.4 Real-time Computational Engine

Multi-dimensional analysis often involves heavy pre-computation and indexing. It becomes more challenging for text-based measures due to their unstructure nature. Moreover, CASeOLAP operations require the computation of neighboring cells (context), hence impose extra complexity. New computation technique needs to be developed to ensure low query latency.

We develop both online and offline computational optimization in TextDive. To better handle the context coupling challenge in multi-dimensional textual space, we develop *utility-based materialization* approaches to achieve better time-space trade-off. Also, several pruning tricks are applied to largely reduce the online computation. We will discuss that in detail in Chapter 6.

CHAPTER 5

SYSTEM DEMONSTRATION

In this chapter, we will demonstrate the detailed design of the system and discuss the use case of the system.

5.1 System Design

In Chapter 1, we show the overal design and work flow of TextDiveIn this section, we will introduce more technical details. The front end page has been shown in Chapter 4. In this section, we will talk more backend design.

In general, the demo system can be divided into three parts, as shown in figure 5.1. On the bottom is the core text OLAP service that performs basic OLAP operations such as cell query, materialization, data import, etc. The CASeOLAP and Document Outlier Detection code also live here. We intentionally seperates the representation layer of the document to achieve deep decoupling, which allows various new representations of a document besides unigram, phrase, sentence and embedding. Also, all the OLAP-like functions such as query and materialization are seperated from the main code base, which allows future researchers to develop their own algorithms or strategies and integrate them into the TextDiveIn the middle is the demo server, which listens to the front-end requests, collects data from the core OLAP service, and then returns the results back to the front-end. On the top is the interactive user interface. It allows the user to specify a query and send it to the demo server. After receiving query results from the demo server, it visualizes them for the user. Here the results are the top representation phrases and sentences, outlier documents, suggested sub-cells, etc. The core OLAP system is implemented using Java $1.8.0_72$. The demo server is implemented using Django $1.9.3$ with Python 2.7. The front end is implemented using HTML5, JavaScript and CSS. The main entrance of our front end page is shown in figure 5.2.

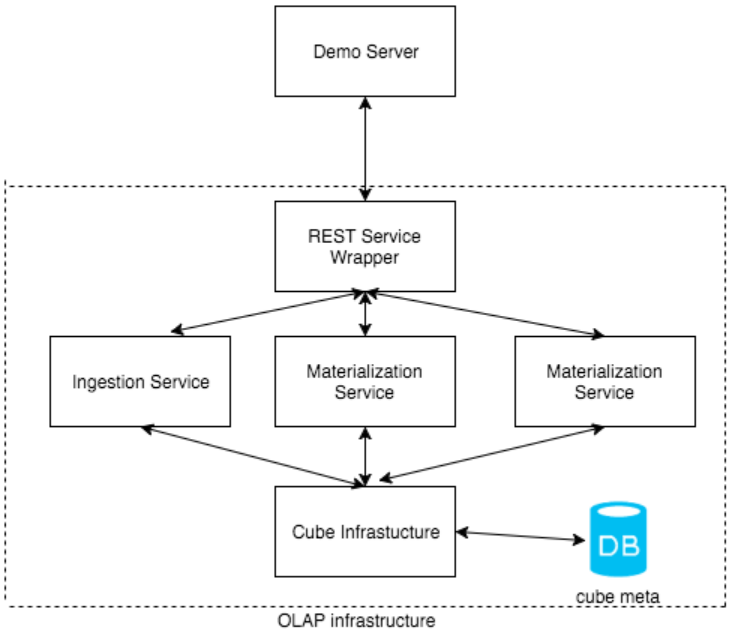

Figure 5.1: TextDive **illustration**

5.2 Use Case: Business Consulting

The use case we demonstrated in this part shows the flexibility and completeness of our prototype as a poweful multi-dimensional text analytical system. Consider the following scenario.

The international policy analyst Tim at McKinsey receives a case from Tesla about what countries the company should expand their solar battery business to, potentially China. He is asked to give a detailed report within one week. First of all, he wants to have a basic understanding of China's major issues that might affect the decision making. What he has on hand is a giant news corpus from New York Times and he is not able to read all of them within one week. So he loads the corpus into TextDive and starts with the query ⟨China⟩, the summarization page gives the top key phrases and key sentences related to the events of China. The intelligent exploration in TextDive then suggest him to explore more in the sub-cells ⟨China, Economy⟩, ⟨China, Environment⟩ and ⟨China, Politics⟩ as those cells contribute the majority information in the ⟨China ⟩cell. So he clicks on the drill down button under the ⟨China, Economy⟩ cell, and TextDive automatically performs the query and returns the summarization of the sub-cell, as shown in figure 5.2. In the result, he finds the government debt has gone up and China's economy has cooled down. More interestingly, China's surplus has risen

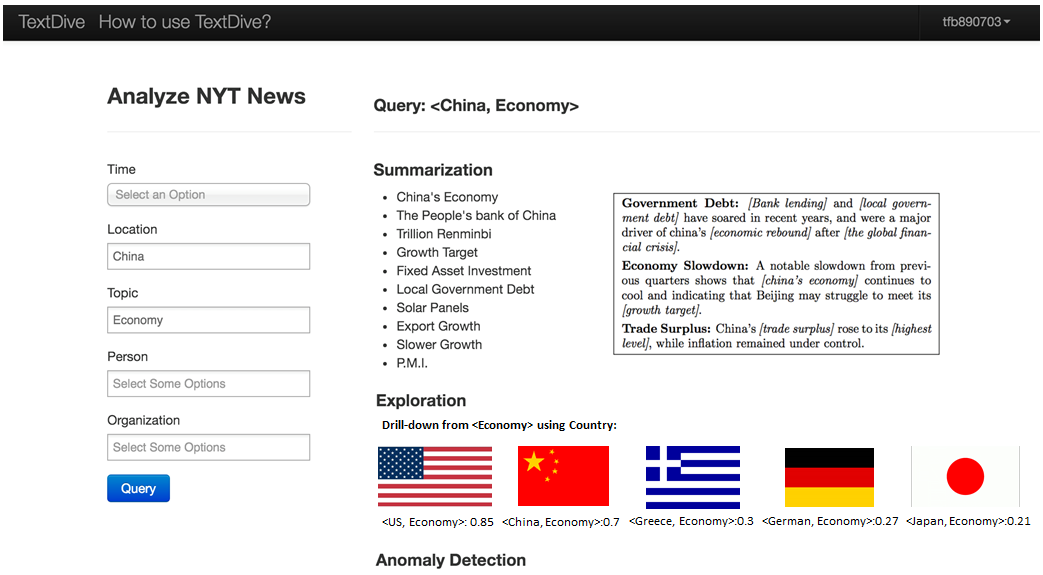

Figure 5.2: TextDive **Screenshot**

to its highest level, while inflation remains under control. Those information are all quite important for the final decision making. He saves them for future reference. He then also explores the other two sub-cells to find the summarization about China's attitude towards environment, especially solar energy, and China's political events, both of which have large effect on whether China could be a good candidate for the business expansion.

In the end, Tim also wants to find out the anomalous events in China that might affect the final decision making so he queries for outlier document for each aforementioned cell. In the cell ⟨China, Economy⟩, he finds something even more interesting. That is, in the past year, there are many lawsuits against China's solar pannel exportation. It is very important information and Tim decides to put it in his report. After finishing the research on China, Tim starts to look for other potential countries for the case. He first queries both ⟨Economy⟩ and ⟨Environment⟩ to check the summarization of documents regarding economic and environmental issues all over the world. Then he uses the intelligent exploration in TextDive to get the recommended list of countries for him to look into. He then does the same research just like what he does with China and finds out the best candidates.

CHAPTER 6

COMPUTATIONAL METHODOLOGY

For any data store, query efficiency is an important issue. This chapter focuses on how to efficiently answer an CASeOLAP query using the ranking measure defined in chapter 4. Note we will only introduce the general idea here, detailed complexiy analysis is provided in [7]. First, Section 6.1 analyzes all the computations needed and presents a framework with partitions of offline and online work. Then, Section 6.2 and 6.3 discuss strategies to reduce the cost for offline and online computation respectively. In our demo system, we implemented all of those strategies for users to choose, according to their requirements and available resources.

6.1 Overview

After applying SegPhrase to generate global phrase candidates and their integrity scores, the following computation tasks are needed for answering each query of cell c: (i) collect a list of candidate phrases (with basic statistics) that appear in cell c, and its sibling cells respectively (ii) compute the popularity and distinctiveness scores for each phrase in cell c, and retrieve their integrity score; (iii) combine the three scores into a single ranking measure, and (iv) sort the phrases and return top-k of them.

Suppose all these computations occur online after a query, the straightforward computation cost is too high to return results timely. The main bottleneck is the first two steps. Computing the neighborhood-aware distinctiveness score requires going through all documents in a target cell as well as in its sibling cells to collect all the statistics. Now suppose we pre-compute them for all the cells, the online query time will be largely reduced to the time of sorting. However, the storage cost will be too high, because the required space is proportional to the sum of the number of unique phrases in each cell over all cells.

Based on these analyses, we partition the online and offline computational work as follows.

1. Generate quality phrase candidates and segmentation of each document using SegPhrase. This is done offline for the entire corpus only once. The integrity score of each phrase is obtained from SegPhrase and stored.
2. Partially compute statistics needed for popularity and distinctiveness score and store them. For certain space-efficient statistics, we fully compute and store them. For other statistics, we selectively compute them for a few cells. This hybrid materialization shifts the most burdensome online computation in step (i) to offline in an economic manner. The offline materialization strategy will be explained in Section 6.2.
3. At online query time, if the target cell has not been fully materialized, generate phrase candidates for the cell, and in the meantime collect their popularity. Use pruning to evaluate distinctiveness of only promising phrase candidates during the top-k phrase generation. This online optimization lowers the cost of step (i)–(ii), and is explained in Section 6.3.

6.2 Hybrid Offline Materialization

What to materialize offline depends on the trade-off between storage cost and online query latency. Nowadays, storage is usually not a hard constraint, while the online analytical query has high demand of low latency. As such, the query latency is given a higher priority. Typically, a well-designed OLAP system should answer every query within a constrained latency. With that constraint satisfied, the lower the storage cost the better.

Following this principle, we design a materialization strategy that can automatically choose what information to materialize according to a given latency constraint $\mathcal{T}$.

The most time-consuming measure to compute is the distinctiveness score, so one natural idea is partial materialization of it. However, it is hard to aggregate because it is not *distributive*. So instead of materializing the score, we reduce the cost of computing it online by saving the cost of collecting statistics in step (i). There are two categories of statistics required for computing distinctiveness:

1. **Phrase-level statistics** $tf(p, c)$ and $df(p, c)$. They are easy-to-aggregate distributive measures.
2. **Cell-level statistics** $cntP(c)$, $cntSib(c)$, $maxDF(c)$ and $avgCP(c)$. $cntSib(c)$ and $avgCP(c)$ are hard to aggregate.

The total number of phrase-level statistics is equal to the total number of distinctiveness and popularity scores, that is $2\lambda \cdot m$, where m is the number of non-empty cells and λ is the average unique phrase count in non-empty cells (*e.g.*, $\lambda = 430.34$ in NYT dataset). The total number of records for cell-level statistics is 4 times m, which is a small fraction of the former (*e.g.*, $4m/(2m * 430) < 0.5\%$). That is to say, materializing the phrase-level statistics has the same cost of materializing the distinctiveness and popularity scores, and materializing cell-level statistics is all affordable.

Based on this observation, we propose the hybrid materialization strategy, where we fully materialize all cell-level statistics and partially materialize the phrase-level statistics.

The rest of this section focuses on how to materialize phrase-level statistics. We first describe how to estimate the time for collecting statistics given a query with a fixed materialization choice, and then present two algorithms for choosing which cells to materialize.

6.2.1 Cost Estimation

In this section, the cost of collecting statistics is measured roughly by the estimated number of CPU clock cycles using the optimal strategy. Although the real runtime can have a large constant factor, the order of magnitude keeps the same. Also, the latency constraint $\mathcal{T}$ has the same unit and is used to compare with the estimated cost.

Among steps (i)–(iv) as we analyzed in Section 6.1, only the cost of step (i) varies with the offline materialization. It also accounts for the most significant part in the query processing time. We can write the total cost of processing each query to cell c as:

$$\mathcal{Q}(c) = \mathcal{Q}_1(c) + \mathcal{Q}_2(c) \tag{6.1}$$

where $\mathcal{Q}_1(c)$ is the cost of step (i) and $\mathcal{Q}_2(c)$ the cost of steps (ii)–(iv). $\mathcal{Q}_2(c)$ can be easily computed for each cell independently with the materialization. So

we focus on the estimation of $\mathcal{Q}_1(c)$, which reduces to estimating the cost of computing $tf(p,\cdot)$ and $df(p,\cdot)$ for cell c and its siblings. Since $tf(p,\cdot)$ and $df(p,\cdot)$ have the shared counting process ($|\mathcal{D}_c|$-way merge join from $|\mathcal{D}_c|$ documents in a cell) and similar aggregation formula, they can be materialized with the same manner and cost. Thus we have:

$$\mathcal{Q}_1(c) = 2 \sum_{c' \in \mathbb{S}(c) \bigcup \{c\}} \mathcal{Q}_{tf}(c') \tag{6.2}$$

where $\mathcal{Q}_{tf}(c')$ is the cost of computing $tf(\cdot, c')$ for cell c'. $\mathcal{Q}_{tf}(\cdot)$ of siblings are included here as sibling statistics are also required for computing representative phrases in cell c.

We show how $\mathcal{Q}_{tf}(c)$ can be recursively estimated in the cell space, for a given cell $c = (a_1, \ldots, a_n, \mathcal{D}_c)$, where $a_i \in \mathcal{A}_i$ (including '$*$'). Without loss of generosity, we assume $des(a_i) \neq \emptyset$ for $1 \leq i \leq n' \leq n$. Thus we have n' aggregation choices; *i.e.*, aggregating cells in one of the following subcell set:

$$S(c)_i = \{c_i = (a_1, \ldots, a, \ldots, a_n, \mathcal{D}_{c_i}) | a \in des(a_i) \wedge \mathcal{D}_{c_i} \neq \emptyset\}$$

Each subcell set $S(c)_i$ of c contains subcells by replacing i-th dimension value to its descendants. Other than aggregating from subcells, one choice is to gather $tf(p,c)$ from raw text. The optimal choice should be used for online computation if the cell is not materialized. Hence, the optimal cost among the $(n'+1)$ choices should be used for our estimation.

As shown by previous work, the OLAP query within the cell space has the *optimal substructure* property. As a consequence, *dynamic programming* can be used for computing optimal cost and choice of aggregation:

$$\begin{aligned}\mathcal{Q}_{tf}(c) = \min \Big\{ & \mathcal{Q}_{raw}(c), \\ & \min_{i:des(a_i)\neq\emptyset} \Big\{ \mathcal{Q}_{agg}(S(c)_i) + \sum_{c' \in S(c)_i} \mathcal{Q}_{tf}(c') \Big\} \Big\}\end{aligned}$$

where $\mathcal{Q}_{raw}(c)$ and $\mathcal{Q}_{agg}(S)$ denote the cost for merging counts from raw text and aggregating from subcell set S respectively. Let λ_c denote the average number of

unique phrases in each document in c, we calculate them as follows.

$$\mathcal{Q}_{raw}(c) = \lambda_c |\mathcal{D}_c| \log |\mathcal{D}_c| \tag{6.3}$$

$$\mathcal{Q}_{agg}(S) = \sum_{c' \in S} |\mathcal{P}_{c'}| \tag{6.4}$$

Eq. (6.3) is obtained by performing a $|\mathcal{D}_c|$-way merge join [10] in the documents contained in cell c. In particular, it scans the sorted phrase lists of documents in parallel. During the merge, $df(p, c)$ can also be counted by the number of lists where p is seen. Equation (6.4) is derived by merging phrase statistics from the subcells in S to the target cell c. Hashmaps are used to guarantee the lookup cost and insertion cost are $\mathcal{O}(1)$.

For a precomputed cell or empty cell, we define:

$$\mathcal{Q}_{tf}(c) = 0 \ \ (c \text{ is materialized or empty}) \tag{6.5}$$

There is one most prominent difference of our query processing cost structure compared with previous OLAP work. The cost for computing neighborhood-aware distinctiveness score for any query is tied to the cost of computation for neighboring cells (siblings in our case), rather than just the target cell. This can be seen from Eq. (6.2). It is a general property for any *neighborhood-aware measure in OLAP*. This new property poses an interesting new challenge to traditional greedy materialization strategy, as the computational cost of sibling cells become *coupled*. We first present an algorithm that ignores this challenge, and then propose a better algorithm to address it.

6.2.2 Simple Greedy Algorithm

We extend the *GreedySelect* algorithm [1] to our task. The algorithm first conducts a *topological sorting* by the *parent-descendant* relationship in the multi-dimensional space. Then it traverses the cells in the bottom-up order. This order ensures that all cells used for aggregating the current cell must have been examined, so the dynamic programming of cost estimation can proceed. For each cell, we estimate the cost with Eq. (6.2) given the currently materialized space. If the cost exceeds the latency constraint $\mathcal{T}$, we materialize the cell c and all its siblings.

This algorithm guarantees that for any online cell query c, the latency is bounded

by a constant. However, the storage cost for the algorithm is more than what is needed. Due to the coupling of cross-sibling computations, the algorithm materializes every sibling of c if its cost exceeds $\mathcal{T}$. In real world multi-dimensional text database, it is common for a cell to have tens or even hundreds of siblings (*e.g.*, cells in *NYT* dataset have 70.7 non-empty siblings on average). In many cases, only part of the siblings need to be materialized to meet the $\mathcal{T}$ requirement. This challenge is specific to measures with dynamic background involved, which cannot be resolved by traditional materialization strategies.

6.2.3 Utility-Guided Greedy Algorithm

We propose a more refined materialization plan, which does not materialize all siblings at once when a cell fails to meet $\mathcal{T}$. Instead, it repeatedly attempts materialization of one sibling, and reevaluates the cost of querying the target cell, until it falls below $\mathcal{T}$. The order of choosing siblings affects how many siblings will be materialized and how much storage cost is needed to meet the constraint. We use a *utility* function for each sibling cell c' to guide this process. Intuitively, we have the following choices of utility function.

1. cost reduction to the target query $\mathcal{Q}_{tf}(c')$;
2. cost reduction to all queries $\mathcal{Q}_{tf}(c')(|\mathbb{S}(c')| + 1)$;
3. cost reduction to critical queries which haven't met the constraint $\mathcal{Q}_{tf}(c')|\{c \in \mathbb{S}(c'), \mathcal{Q}(c) \geq \mathcal{T}\}|$; and
4. cost reduction to all queries per storage unit $|\mathbb{S}(c')|$;
5. cost reduction to critical queries per storage unit $|\{c \in \mathbb{S}(c'), \mathcal{Q}(c) \geq \mathcal{T}\}|$.

The cost reduction to the target query per storage unit is a constant 1, which cannot provide any guidance.

The choices 2–5 all reflect the cost reduction beyond the target query. Due to the neighborhood coupling, the computational benefit of a particular cell is shared by neighboring cells, *i.e.*, siblings in our task. Since the sibling relationship is mutual (c's siblings must have c as sibling as well), the pre-computation of c' reduces the cost querying siblings of c', and querying itself. Hence we have the factor $(|\mathbb{S}(c')| + 1)$ in choice 2. Choice 3 is similar, except that it values the cost reduction only to the queries that currently cannot be answered within time $\mathcal{T}$. Choice 4 and 5 normalize the cost reduction by the storage cost of materialization,

which measures the unit gain. This refined version may requires to monitor $\mathcal{Q}(\cdot)$ of unexamined cells to compute the utility function. According to the definition of sibling, the siblings of cell c share the same cuboid of c. Therefore we cope with this by grouping non-empty cells into cuboids and estimate $\mathcal{Q}_{tf}(\cdot)$ and $\mathcal{Q}_1(\cdot)$ of all cells in the cuboids before materializing any of them. In the concrete algorithm, one of the five *utility* functions is used to provide different balance between query time and storage.

The utility-guided algorithm also guarantees the latency requirement. In the experiments, we show that utility-guided algorithm can reduce the storage cost with the same time latency. We also compare the overall space efficiency of various utility choices.

6.3 Optimized Online Processing

The vanilla online processing needs to compute the ranking measure for all phrase candidates in a cell in order to sort them. The computation of the distinctiveness score can be expensive, if the cell is not materialized. We propose an early termination and skipping technique to prune phrase candidates that are impossible to be among top-k.

Our technique is based on two facts. First, the distinctiveness score is the only more expensive measure to compute than phrase candidate generation. This inspires us to decompose the overall ranking measure into two parts: the part that relies on distinctiveness score, and the part not. The latter part $pop(p, c) \cdot int(p)$ can be computed for each phrase candidate cheaply. Second, the range of the two parts are both between 0 and 1. That indicates the overall ranking score is bounded by $pop(p, c) \cdot int(p)$. In fact, if we can estimate a more accurate upper bound of $disti(p, c)$, we can also derive a tighter bound for the overall ranking score, and largely prune the phrase list.

We first sort all phrase candidates by $u_1(p, c) = pop(p, c) \cdot int(p)$, and go through them one by one. That is, phrases with high cell popularity and integrity get evaluated early. As soon as the next phrase p has a lower u_1 than the lowest final score θ of phrases in the top-k list, it is safe to terminate the enumeration. Otherwise, we estimate a tighter upper-bound $u_2(p, c)$ without using siblings' phrase-level statistics.

$$u_2(p, c) = \frac{e^{rel(p,c)}}{1 + e^{rel(p,c)}} \tag{6.6}$$

u_2 only relies on $rel(p, c)$ which can be computed by cell-level statistics and the phrase p's frequency and document frequency in the current cell (Eq. (6.6)). Since the cell-level statistics are fully materialized, and $tf(p, c)$ is already obtained when computing $pop(p, c)$, the calculation of $u_2(p, c)$ only incurs one aggregation of $df(p, c)$. If $u_1(p, c) \cdot u_2(p, c) < \theta$, we can skip the actual computation of distinctiveness score and move on to the next candidate. In the worst case, we have to retrieve sibling statistics, which involves aggregations for non-materialized sibling cells.

CHAPTER 7

CONCLUSION

In this demo, we build an end-to-end, real-time analytical system TextDive to systematically handle a collection of text data. We build a multi-dimensional textual space by combining extracted typed entities and existing meta-attributes. We also implement three OLAP-like operations to support summarization, exploration and anomaly detection in multi-dimensional textual space. TextDive aims to empower analysts to load massive text corpora, manage them with multiple dimensions and gain insightful knowledge by posing interactive queries.

REFERENCES

[1] C. X. Lin, B. Ding, J. Han, F. Zhu, and B. Zhao, "Text cube: Computing IR measures for multidimensional text database analysis," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, 2008. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2008.135 pp. 905–910.

[2] A. Simitsis, A. Baid, Y. Sismanis, and B. Reinwald, "Multidimensional content exploration," *PVLDB*, vol. 1, no. 1, pp. 660–671, 2008. [Online]. Available: http://www.vldb.org/pvldb/1/1453929.pdf

[3] D. Zhang, C. Zhai, J. Han, A. Srivastava, and N. Oza, "Topic modeling for olap on multidimensional text databases: Topic cube and its applications," vol. 2, no. 56. New York, NY, USA: John Wiley & Sons, Inc., Dec. 2009. [Online]. Available: http://dx.doi.org/10.1002/sam.v2:5/6 pp. 378–395.

[4] F. Tao, B. Zhao, A. Fuxman, Y. Li, and J. Han, "Leveraging pattern semantics for extracting entities in enterprises," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15. New York, NY, USA: ACM, 2015. [Online]. Available: http://doi.acm.org/10.1145/2736277.2741670 pp. 1078–1088.

[5] X. Ren, A. El-Kishky, C. Wang, F. Tao, C. R. Voss, and J. Han, "Clustype: Effective entity recognition and typing by relation phrase-based clustering," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: ACM, 2015. [Online]. Available: http://doi.acm.org/10.1145/2783258.2783362 pp. 995–1004.

[6] J. Liu, J. Shang, C. Wang, X. Ren, and J. Han, "Mining quality phrases from massive text corpora," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. New York, NY, USA: ACM, 2015. [Online]. Available: http://doi.acm.org/10.1145/2723372.2751523 pp. 1729–1744.

[7] "Context-aware semantic olap in text cubes," *In submission*, 2016.

[8] F. Tao, K. H. Lei, J. Han, C. Zhai, X. Cheng, M. Danilevsky, N. Desai, B. Ding, J. G. Ge, H. Ji, R. Kanade, A. Kao, Q. Li, Y. Li, C. Lin, J. Liu, N. Oza, A. Srivastava, R. Tjoelker, C. Wang, D. Zhang, and B. Zhao, "Eventcube: Multi-dimensional search and mining of structured and text data," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013. [Online]. Available: http://doi.acm.org/10.1145/2487575.2487718 pp. 1494–1497.

[9] F. Tao, G. Brova, J. Han, H. Ji, C. Wang, B. Norick, A. El-Kishky, J. Liu, X. Ren, and Y. Sun, "Newsnetexplorer: Automatic construction and exploration of news information networks," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: ACM, 2014. [Online]. Available: http://doi.acm.org/10.1145/2588555.2594537 pp. 1091–1094.

[10] S. Bedathur, K. Berberich, J. Dittrich, N. Mamoulis, and G. Weikum, "Interesting-phrase mining for ad-hoc text analytics," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 1348–1357, Sep. 2010. [Online]. Available: http://dx.doi.org/10.14778/1920841.1921007

[11] A. Inokuchi and K. Takeda, "A method for online analytical processing of text data," in *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, 2007. [Online]. Available: http://doi.acm.org/10.1145/1321440.1321506 pp. 455–464.

[12] F. Ravat, O. Teste, R. Tournier, and G. Zurfluh, "Top_keyword: An aggregation function for textual document olap," in *Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery*, ser. DaWaK '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 55–64. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85836-2_6

[13] B. Zhao, X. Lin, B. Ding, and J. Han, "Texplorer: Keyword-based object search and exploration in multidimensional text databases," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, ser. CIKM '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2063576.2063822 pp. 1709–1718.

[14] B. Ding, B. Zhao, C. X. Lin, J. Han, and C. Zhai, "Topcells: Keyword-based search of top-k aggregated documents in text cube," in *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, 2010. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2010.5447838 pp. 381–384.

[15] M. Mendoza, E. Alegría, M. Maca, C. Cobos, and E. León, "Multidimensional analysis model for a document warehouse that includes textual measures," *Decision Support Systems*, vol. 72, pp. 44–59, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.dss.2015.02.008

[16] M. A. Hearst, "Clustering versus faceted categories for information exploration," *Commun. ACM*, vol. 49, no. 4, pp. 59–61, 2006. [Online]. Available: http://doi.acm.org/10.1145/1121949.1121983

[17] D. Tunkelang, *Faceted Search*, ser. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2009. [Online]. Available: http://dx.doi.org/10.2200/S00190ED1V01Y200904ICR005

[18] O. Ben-Yitzhak, N. Golbandi, N. Har'El, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. J. Shekita, B. Sznajder, and S. Yogev, "Beyond basic faceted search," in *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008, Palo Alto, California, USA, February 11-12, 2008*, 2008. [Online]. Available: http://doi.acm.org/10.1145/1341531.1341539 pp. 33–44.

[19] D. Dash, J. Rao, N. Megiddo, A. Ailamaki, and G. M. Lohman, "Dynamic faceted search for discovery-driven analysis," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, 2008. [Online]. Available: http://doi.acm.org/10.1145/1458082.1458087 pp. 3–12.

[20] T. Baldwin and S. N. Kim, "Multiword expressions," in *Handbook of Natural Language Processing, Second Edition.*, 2010, pp. 267–292. [Online]. Available: http://www.crcnetbase.com/doi/abs/10.1201/9781420085938-c12

[21] Z. Zhang, J. Iria, C. Brewster, and F. Ciravegna, "A comparative evaluation of term recognition algorithms," in *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*, 2008. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2008/summaries/538.html

[22] M. Danilevsky, C. Wang, N. Desai, X. Ren, J. Guo, and J. Han, "Automatic construction and ranking of topical keyphrases on collections of short documents," in *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, 2014. [Online]. Available: http://dx.doi.org/10.1137/1.9781611973440.46 pp. 398–406.

[23] A. El-Kishky, Y. Song, C. Wang, C. R. Voss, and J. Han, "Scalable topical phrase mining from text corpora," *PVLDB*, vol. 8, no. 3, pp. 305–316, 2014. [Online]. Available: http://www.vldb.org/pvldb/vol8/p305-ElKishky.pdf

[24] R. C. Wang and W. W. Cohen, "Language-independent set expansion of named entities using the web," in *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, 2007. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2007.104 pp. 342–350.

[25] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1879

[26] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open information extraction from the web," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, ser. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1625275.1625705 pp. 2670–2676.

[27] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Web-scale information extraction in knowitall: (preliminary results)," in *Proceedings of the 13th International Conference on World Wide Web*, ser. WWW '04. New York, NY, USA: ACM, 2004. [Online]. Available: http://doi.acm.org/10.1145/988672.988687 pp. 100–110.