Dynamic multi-dimensional models for text warehouses

Maria Zamfir Bleyberg, Karthik Ganesh
Computing and Information Sciences Department
Kansas State University, Manhattan, KS, 66506

Abstract

In this paper, we introduce a *dynamic multi-dimensional model*, which is suitable for building text warehouses. The dimensions are *atomic semantic categories* embedded in a familiar taxonomy. This approach to text warehouses requires a large number of dimensions, some of which may be not known in advance. Central to the dynamic multi-dimensional model is the *meta-snowflake schema*, which is a snowflake schema with an index table. The index table contains *metadata* on dimensions consisting of atomic and compound semantic categories. The documents stored in the warehouse are retrieved according to the semantic categories assigned to them. Such a text warehouse increases the precision and efficiency of document exploration.

Keywords: document exploration, compound semantic categories, text warehouses, snowflake schemas.

1 Introduction

In recent years, we have witnessed an immense growth in the availability of on-line information. Given that much of this information is textual in nature, complex paradigms for knowledge discovery in text have been developed to retrieve *only* those documents that are requested by the user. Simple text categorization [5], text categorization using keyword graphs and background knowledge [4, 6, 3], and categorial logical systems using syntactic categories and their semantic counterparts [9, 11] are examples of methodologies for classifying documents.

The organization of documents in traditional or virtual warehouses also increases the efficiency of document exploration. In the broadest sense, a data warehouse refers to a single, subject-oriented, integrated, time-variant collection of data that supports the analytical/decision-making functions in most organizations [7]. A very important component of a data warehouse, which is critically important to successful implementations, is the metadata repository. A metadata repository is a database that describes the characteristics of the data and the environment in which the data are managed in the warehouse. Creating metadata directly in a database and linking it to resources is growing in popularity, in particular, to develop subject-based gateways for documents.

In this paper, we introduce a *dynamic multi-dimensional model* and use it to build a text warehouse prototye with efficient document retrieval capabilities. This work is based on *atomic* and *compound semantic categories* for document exploration, which have been introduced in [1, 2]. The dimensions of the dynamic multi-dimensional model are atomic semantic categories embedded in a familiar taxonomy. There is a large number of semantic categories, many of which may not be known in advance. The typical multi-dimensional storage model, which is used to design data warehouses, has a limited representational power; the fixed number of dimensions, which must be known in advance, makes it unsuitable for building text warehouses. The use of atomic semantic categories as dimensions of a text warehouse calls for a multi-dimensional model that supports the addition of new dimensions to an already created text warehouse.

The dynamic multi-dimensional model supports a scaleable architecture that can handle increased demand of dimensions as needed. Central to the dynamic multi-dimensional model is the *meta-snowflake schema*, which is a snowflake schema with an index table. The index table contains metadata on dimensions consisting of atomic and compound semantic categories that label the documents stored in the warehouse. The index table can be expanded when a document with a new semantic category must be added to the warehouse.

2 Atomic and compound semantic categories

Following the idea presented in Sommers' sense logic [10], in [1, 2], we introduced a binary *sense type decision tree* having *atomic sense types* (i.e., *atomic semantic categories*) as nodes, which is used to derive *compound semantic categories* under given composition rules. The composition rules capture the *mean-*

[*a*] *Oracle hires grads to build a warehouse with a green roof:*	(**Company** $\Rightarrow$ **Person**) $\Rightarrow$ **Building**
[*b*] *Oracle hires grads to build a data warehouse:*	(**Company** $\Rightarrow$ **Person**) $\Rightarrow$ **Application**
[*c*] *Oracle and Compaq developed a joint training program:*	**Company** $\Rightarrow$ **Concept**
[*d*] *Oracle and Compaq developed programs for desktops:*	**Company** $\Rightarrow$ **Application**

Figure 1: English sentences and their compound semantic categories

ingful relationships among semantic categories. We distinguish two kinds of sentences: *meaningful* ones and *meaningless* ones. An adjective or verb associated with a noun generates either a *meaningful* sentence or a *meaningless* sentence. As a result, a universe of English nouns from a given collection of documents is associate to the root of the sense type decision tree. The role of decision tree attributes is played by adjectives and verbs.

A binary sense type decision tree *does not* represent the classification of words in various categories, such as, *green*, *red*, and *yellow* are *colors*. It shows the nouns and pronouns partitioned into complementary sets in such a way that certain adjectives and verbs can be meaningfully applied to all the nouns of a group and cannot be meaningfully applied to the nouns of the other group. Some nouns, such as *program*, may belong to several categories.

An example of a binary sense type decision tree (with some instances) is given in Figure 2. This decision tree is used to assign semantic categories to the sentences in Figure 1. In this decision tree, we have the node **Product** associated with the set $\{program^1, warehouse^1,$ *workstation, desktop*$\}$. The verb *run* in conjunction with $program^1$ and $warehouse^1$ leads to **meaningful** sentences: *a* $program^1$ *runs* [*on a computer*], *a* [*data*] $warehouse^1$ *runs* [*on a mainframe*] (see sentences [*b*] and [*d*]). The same verb leads to **meaningless** sentences in conjunction with *workstation* and *desktop*, such as, *a workstation runs*. Some nouns have several meanings. In such a case, the noun will be indexed, as in a dictionary. For example, we write $program^1$ if *program* denotes a computer program, and $program^2$ if *program* denotes a plan to be followed.

Sense types are defined as follows:

- each atomic semantic category is a sense type,
- if α and β are sense types, then $(\alpha \rightarrow \beta)$ is a sense type,
- if $\alpha \rightarrow \beta$ is a sense type, then $(\alpha \Rightarrow \beta)$ is a sense type; $(\alpha \Rightarrow \beta)$ is a *compound semantic category.*

An atomic semantic category denotes the class of nouns it can label. The connective $\Rightarrow$ makes it possible to explicitly refer to compound semantic categories.
Each type $(\alpha \rightarrow \beta) \rightarrow (\alpha \Rightarrow \beta)$ is intended to denote some given set of functions from the type α and the type β to $(\alpha \Rightarrow \beta)$. The interpretation of a compound semantic category is a collection of *meaningful* sentences. A sentence is *meaningless* if no semantic category can be assigned to it.

Underlying the interpretation of a compound semantic category is the concept of a *typed lambda abstraction.* Lambda abstraction [8] has proven useful in writing *function expressions* and *application*, which allows one to make use of the functions defined. A simple example of a typed lambda expression is

$$I : \mathbf{nat} \rightarrow \mathbf{nat} \;\equiv\; (\lambda x^{\mathbf{nat}} \,.\, x^{\mathbf{nat}})^{\mathbf{nat} \rightarrow \mathbf{nat}}$$

representing the identity function on natural numbers.

Examples of how verbs and adjectives can be represented as sense typed lambda expressions are given below.

The transitive verb *hire* could be represented by the sense typed lambda expression

$hire$: **Company**$\rightarrow$**Person**$\rightarrow$(**Company**$\Rightarrow$**Person**)

This lambda expression shows that *hire* is a function, such that, when the first argument is a noun of type subsumed by **Company** as an instance and the second argument is a noun of type subsumed by **Person** as an instance, a meaningful expression of compound type (**Company** $\Rightarrow$ **Person**) results. **Company** and **Person** are the types of the *largest* categories of nouns for which it makes sense to use the verb *hire* (the highest node in the sense type decision tree in Figure 2).

An instance of a meaningful verb expression is

$$hire(Oracle, grads) : \ \mathbf{Company} \Rightarrow \mathbf{Person}$$

i.e., *Oracle hires grads.*

We choose to use a prefix notation for functions, because this representation eliminates the variations of the locations of words in a sentence.

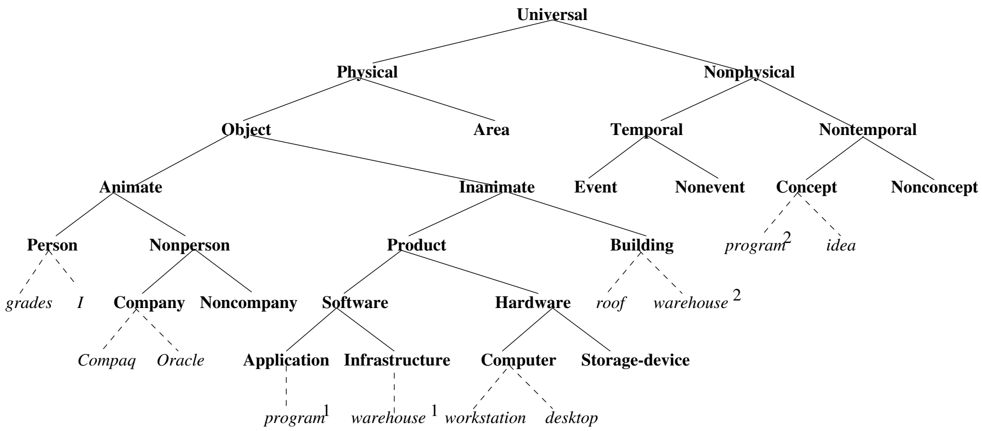

Figure 2: A sense type decision tree with instances

A verb expression such as

$$hire(Compaq, program^1)$$

is meaningless, because *program*^{1} is of type **Application**; **Application** and **Person** are on different branches in the sense type decision tree in Figure 2.

Adjectives are represented in a similar way. The adjective *green* is represented by the sense typed lambda expression

$$green : \textbf{Physical} \rightarrow \textbf{Physical}$$

This lambda expression shows that *green* is a function of type **Physical** → **Physical**, such that, when the argument is a noun of type subsumed by **Physical** as an instance, a noun phrase of type **Physical** is produced. As a result, the following adjective sense type lambda expression

$$green(roof) : \textbf{Physical}$$

is meaningful, where-as the adjective expression

$$green(ideas)$$

is meaningless, because *idea* is of type **Concept**; **Concept** and **Physical** belong to different branches in the sense type decision tree in Figure 2.

In this paper, we assume that the semantic category of every document that must be included in the warehouse is already known. The assignment of semantic categories to documents is covered in [2].

3 Meta-snowflake schemas

W.H. Inmon [7] characterized a data warehouse as "a subject-oriented, integrated, nonvolatile, time-variant collection of data in support of management's decisions". To ensure easy access to this vast amount of data, designers of modern data warehouses typically adopt a dimensional approach to information processing instead of a traditional relational database approach. In this model, data is divided into two categories: *facts* and *dimensions*. Facts are the core data elements being analysed, and dimensions are attributes about facts. This formalism of representing data is known as *star schema*. The facts are represented as a table in the center of the schema. It is the only table in the schema with multiple joins connecting it to the dimension tables. Dimensions with hierarchies are often decomposed into *snowflake* structures.

A typical data warehouse has the following components:

- Data migration tools, which access the source data and transform it.
- Metadata repositories that describe the data warehouse.
- A warehouse data store that provides rapid access to the data.
- A collection of tools for retrieving, formatting, and analyzing the data.
- Tools for managing the warehouse environment.

A metadata repository is a database that describes the characteristics of the data and the environment in which the data are managed in the warehouse.

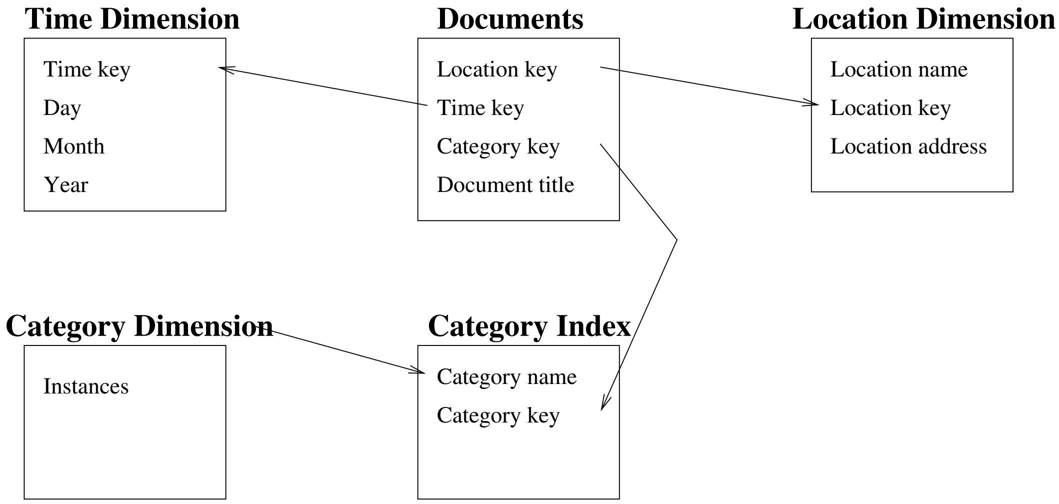

Figure 3: The meta-snowflake schema of the text warehouse prototype

In this paper, we focus on the development of a text warehouse prototype with efficient document retrieval capabilities. The dimensions of the model for the text warehouse are atomic semantic categories embedded in a sense type decision tree. There is a large number of semantic categories, many of which may not be known in advance. The typical multi-dimensional storage model, which is used to design data warehouses, has a limited representational power; the fixed number of dimensions, which must be known in advance, makes it unsuitable for building text warehouses. The description of dimensions is part of the metadata repository. Therefore, in a text warehouse, not just the data but the metadata keeps on changing. Our goal is to capture the dynamics of the dimensions in the conceptual model. We accomplish this task by introducing a *meta-snowflake schema*, which is a snowflake schema with a *category index table* added to it. The index table contains metadata on dimensions, which are atomic semantic categories, and compound semantic categories, which label the documents stored in the warehouse.

Many dimensions have hierarchical relationships, which are imposed by the existing paths in the sense type decision tree. For example, according to the sense type decision tree in Figure 2, we have:

- **Application** *is* **Software** *is* **Product**
- **Computer** *is* **Hardware** *is* **Product**

In Figure 3 we give an example of a meta-snowflake schema for a text warehouse, which has the fact table **Documents** and the index table **Category Index** that supports a variable number of dimensions. **Time** and **Location** are typical data warehouse dimensions. Traditional data warehouses employ a supply-driven view of the information resources, where-as virtual data warehouses employ a demand-driven view of the information resources. In this work, we consider a virtual text warehouse prototype. A document, which is a core element of the fact table, is represented by title and location key. The document is actually stored at the location address. Category names represent semantic categories, which have the role of delivering subject-based documents with precision and efficiency. The documents of the present text warehouse prototype are web sample news from Yahoo related to various hi-tech companies.

4 Functions of the text warehouse prototype

The present text warehouse prototype has two major functions: to update the category index table and to retrieve documents by using semantic categories. The prototype provides a graphical user interface (GUI) only for document retrieval (see Figure 5) and another one for modification of the category index table (see Figure 6). Any user can browse the text warehouse, but only the warehouse administrator can modify the category index table.

Figure 5 displays the GUI for the retrieval of documents by using semantic categories. The example shown is a request for documents having the compound semantic category **Company** $\Rightarrow$ **Person** $\Rightarrow$ **Product**, where Company is Oracle. Figure 6 displays the GUI for the modification of category index table. The menu for “Add” consists of “Dimension” and “Category”. “Dimension” is used when a new dimension, which is an atomic semantic category, must

Time Dimension

Day	Month	Year	Time key
22	01	2000	t001
23	02	2000	t002

Person Dimension

Instances
Bill Gates
Larry Ellisons
grads

Company Dimension

Instances
Oracle
Microsoft
Cisco

Category Index

Category name	Category keys
Company	c001
Person	c002
Building	c003
Application	c004
Product	c005
Company => Person	c006
Company => Product	c007
Company => Person => Product	c008

Application Dimension

Instances
warehouse 1

Building Dimension

Instances
warehouse 2
roof

Figure 4: Relations corresponding to meta-snowflake schema tables

be added to the index table. “Category” is used when a compound semantic category must be added to the index table. All the atomic semantic categories that form the compund semantic category must already be in the index table.

We chose the Oracle database management system on a UNIX platform to implement the text warehouse database and the Java object-oriented programming language with embedded SQL to implement the graphical user interfaces.

5 Summary and Future Work

In this paper, we described a dynamic multidimensional model that is suitable for the design of text warehouses.

Work in progress includes the extension of the current prototype with functions that support OLAP queries, which can reveal interesting trends regarding the document exploration.

References

[1] M.Z. Bleyberg. Preserving text categorization through translation. In *Proc. 1999 IEEE International Conference on Systems, Man, and Cybernetics*, 1999.

[2] M.Z. Bleyberg. Sense type decision trees for natural language processing. In *Proc. 12th Int. Conf. on Control Systems and Computer Science*, 1999.

[3] R. Feldman and H. Hirsh. Mining associations in text in the presence of background knowledge. In *KDD'96 - Proc. 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, 1996.

[4] U. Hahn and K. Schnattinger. Deep knowledge discovery from natural language texts. In *KDD'97 - Proc. 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 175–178, 1997.

[5] U. Hahn and K. Schnattinger. Knowledge mining from textual sources. In *CIKM'97 - Proc. 6th Intl. Conf. on Information and Knowledge Management*, pages 83–90, 1997.

[6] M. Hearst and C. Karadi. Searching and browsing text collections with large category hierarchies. In *Proc. of the ACM SIGCHI Conf. on Human factors in Computing Systems (CHI)*, 1997.

[7] W. H. Inmon. *Building the Data Warehouse.* John Wiley and Sons, Inc, 1996.

[8] John Mitchell. *Foundations for Programming Languages.* The MIT Press, 1996.

[9] Michael Moortgat. Categorial type logics. In *Handbook of Logic and Language*, 1998.

[10] Fred Sommers. Types and ontology. *Philosophy Review*, (72), July 1963.

[11] Raymond Turner. Types. In *Handbook of Logic and Language*, 1998.

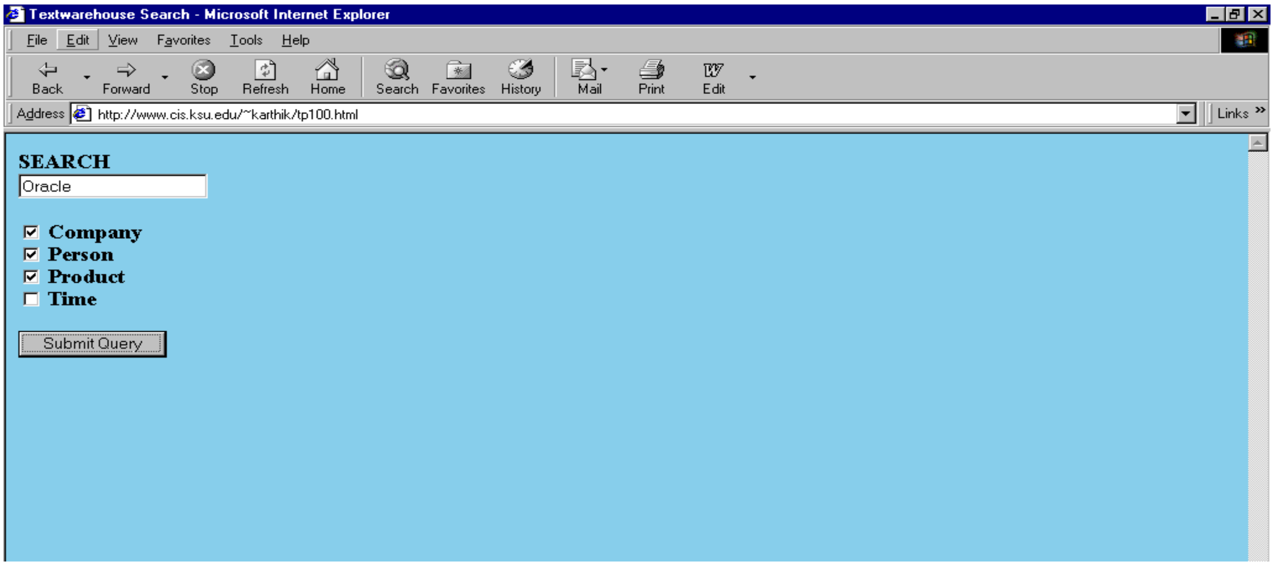

Figure 5: GUI for the retrieval of documents

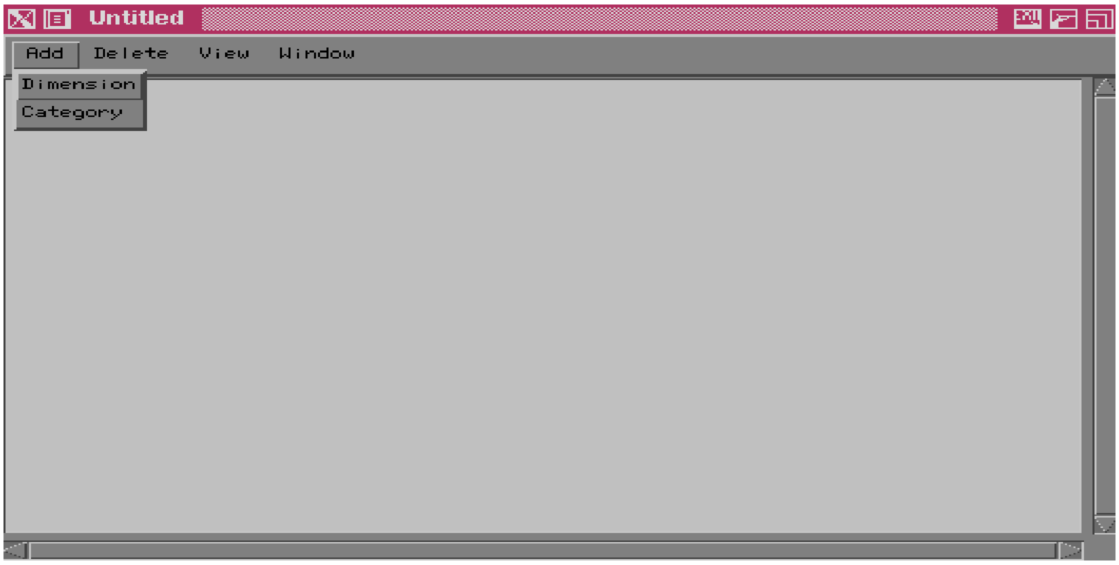

Figure 6: GUI for modification of the category index table