A Robust System Architecture for Mining Semi-structured Data

Lisa Singh, Bin Chen, Rebecca Haight, Peter Scheuermann, Kiyoko Aoki

Northwestern University
2145 Sheridan Road
Evanston, IL 60208
{lsingh, bchen, rhaight, peters, kaoki}@ece.nwu.edu

Abstract

The value of extracting knowledge from semi-structured data is readily apparent with the explosion of the WWW and the advent of digital libraries. This paper proposes a versatile system architecture for text mining that maintains structured data components in a relational database and unstructured concepts in a concept library. After a detailed explanation of our system architecture, we briefly describe IRIS, our prototype rule generation system

Introduction

Although much attention has been given to extracting knowledge from structured data, more and more tools that extract knowledge from semi-structured data are becoming available. The shift in focus is due in large part to the explosion of the World Wide Web (WWW) and the advent of digital libraries. Data from these arenas is potentially an invaluable source for analysis and decision support.

The success of a text mining tool is dependent upon the ability to accurately represent document content and efficiently generate rules. This paper proposes a scalable and versatile system architecture for text mining that provides us with the infrastructure necessary to accomplish these goals.

Figure 1 illustrates the high-level system architecture proposed in this paper. The rule generator is the central component in the design. It parses a request submitted by the user and determines an execution strategy based on specified constraints. The strategy indicates the retrieval order and the amount of data requested from the concept library and the database. If necessary, additional up-to-date information is requested from the information discovery module. The rule generator processes all this information and generates rules (association, classification, etc.) in response to user requests.

This system design has a number of advantages. First, a distinction is made between structured attributes that occur regularly across a document set and terms or phrases that are less structured and occur unpredictably. The former is placed in the database, while the latter is stored in the concept library. Distinguishing these two types of data allows us to represent them more accurately, thereby giving us the ability to generate more interesting rules. Second, the concept library differentiates concepts associated with different domains. This in turn implies that the final set of rules is semantically more accurate than those systems that do not maintain a concept structure or provide only a single ontology. Third, since only a 'meaningful' subset of information from a document is stored in the concept library and the database, our document representation is more compact than storing the complete document locally. Next, for static domains where the content of documents remains constant over time, we need only parse the local or remote documents once off-line. Consequently, the execution time of the parsing algorithm does not affect the user response time. For more dynamic domains, document content may change. In this case, only documents that have changed recently need to be parsed again. Finally, any database, e.g. relational, object-oriented, semi-structured, can be used as the database component illustrated in Figure 1. Therefore, the data mining architecture presented here can be built above existing databases. Once the database returns data about the more structured components of the dataset, the semi-structured data mining tool uses it in conjunction with concept knowledge attained from the concept library to efficiently generate inductive rules.

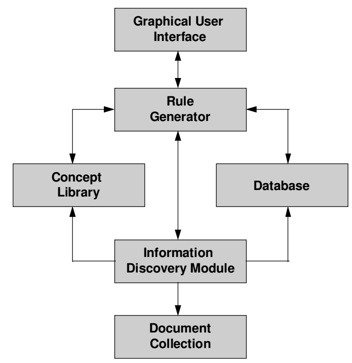

Figure 1: System Architecture

The remainder of this paper is organized as follows. Section 2 presents some background concepts and related literature. In section 3, our semi-structured data mining tool architecture is proposed and discussed in detail. Section 4 briefly describes IRIS, our prototype rule generation system built according to this system architecture. Finally, section 5 presents our conclusions.

Background Concepts & Related Literature

Magazine articles, research papers, and World Wide Web HTML pages are traditionally considered semi-structured data. Each of these examples contains some clearly identifiable features, e.g. author, date, publisher, WWW address. In this paper, we refer to these identifiable features as *structured attributes* or *structured objects*, depending on the type of database the features are stored in. Each document also includes blocks of text that are considered unstructured components of the document, e.g. abstract, headings, and paragraphs. We define an *unstructured concept* to be any meaningful word, phrase, acronym, or name extracted from these unstructured blocks of text.

Many types of rules can be created using semi-structured data. To date, researchers have focused on generating association, trend, classification, and clustering rules from text (Amir, Feldman, and Kashi 1997; Feldman and Hirsh 1996; Lent, Agrawal, and Srikant 1997; Lagus et al. 1996; Singh, Scheuermann, and Chen 1997; Tresch, Palmer, and Luniewski 1995). One of the distinctions between rule generation algorithms for structured data and semi-structured data involves the need to add concept information into a rule. For example, association rules are of the form $A \Rightarrow B$. Within the structured domain, (Agrawal, Imielinski, and Swami 1993) defines both A and B to be sets of structured values for a single attribute. Within the semi-structured domain, (Singh, Scheuermann, and Chen 1997) define both A and B to be sets of unstructured concepts and structured values within and across attributes.

Because of space limitations, we cannot go through detailed examples of each rule type. We refer the readers to papers focusing on semi-structured rule generation algorithms (Amir, Feldman, and Kashi 1997; Feldman and Dagan 1995; Feldman and Hirsh 1996; Lent, Agrawal, and Srikant 1997; Singh, Scheuermann, and Chen 1997). Instead we briefly identify four considerations unique to the semi-structured domain that should be addressed when designing a data mining architecture.

- Unstructured concepts exist within documents and therefore, should be represented in a data model.
- Semantic knowledge about relationships among concepts is important, but costly as an online process.
- Since background knowledge can exist for structured or unstructured data, a system should be flexible enough to store multiple formats of background knowledge.
- Searching the document space for all combinations of structured value – concept pairs requires exponential time. Therefore, within the semi-structured domain providing users with a mechanism for constraining the search space is the norm.

These considerations associated with the semi-structured domain support our claim that a specialized architecture for semi-structured data mining is useful. To date, we are unaware of any other specialized architectures proposed for semi-structured data mining applications.

Proposed Architecture

There are five major components in the architecture presented in Figure 1: the graphical user interface (GUI), the rule generator, the concept library, the database, and the information discovery module. The *GUI* allows the user to specify attributes and concepts he wants to use as constraints when generating rules. The *rule generator* processes this information, sends data requests to the concept library and the database, and determines a meaningful set of rules. The *database* contains the structured data values extracted from the text, as well as mappings between tuples and documents. The *concept library* maintains general and specialized unstructured concepts, relationships among concepts, and mappings between concepts and documents. The *information discovery module* extracts concepts and structured values from a document collection and updates the database and the concept library as necessary. The remainder of this section details the features of each of these components.

Concept Library Architecture

The *concept library* consists of three components: the *concept dictionary*, the *specialized concept guides,* and the *concept indices*. The *concept dictionary* contains common concepts extracted from a basic dictionary. In contrast, each *specialized concept guide* maintains domain specific concepts. Both the concept dictionary and the specialized concept guide maintain relationships among concepts. Finally, the *concept indices* maintain lists of document ids associated with each concept in the concept library. Figure 2 shows the relationships among these three components.

Our concept library architecture distinguishes general concepts from specialized concepts associated with a particular domain, e.g. computer science document collection, on-line coffee shops or wineries. As an example, suppose our domain is on-line coffee shops. Then a general or generic concept may be *coffee bean*, while *latte* could be considered a specialized concept.

(Singh 1997) found that a large percentage of the useful concepts appearing within a document collection are part of a common set. This common set involves concepts with semantic relationships that appear in an ordinary dictionary.

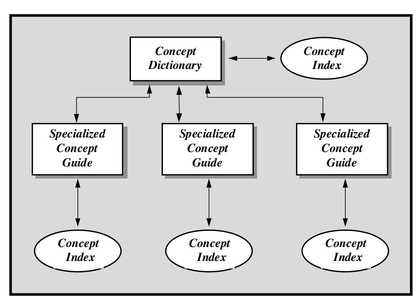

Figure 2: Concept Library Components

Therefore, by maintaining the concept dictionary, we can eliminate the process of regenerating this common set or rediscovering relationships among concepts every time a new domain is added to our system. For each set of related concepts, the relationship type is also maintained. Example relationship types include: broad-narrow (Parent-child), related (sibling), synonym, antonym, and part-to-whole.

Important domain specific concepts exist outside of this common set and are, therefore, not included in the concept dictionary. For example, in a medical database, disease names can be very specialized. Also, in some cases a concept that exists in the concept dictionary may have a different semantic meaning within a specialized document collection. In the concept dictionary, the concept *thread* refers to a thin fibrous strand, while in the computer science domain, the same concept refers to a mini process. We must be able to account for these differences across domains. Therefore, our specialized concept guide maintains domain specific concepts and relationships. The concepts in a specialized concept guide include terms, phrases, proper names, and acronyms. (Only terms and phrases exist in the concept dictionary.)

A concept index identifies associations between concepts and documents by mapping each concept to a set of documents that contain the concept. As illustrated in Figure 2, a separate concept index is maintained for the concept dictionary and each specialized concept guide.

Database

Because structured attributes occur regularly across documents within a domain, each one is included in the database schema. Determining this schema for every domain is a preprocessing step. (Meaningful information not represented in the database is captured in the concept library.) Once this information is extracted from a document, rules can be generated using structured attribute values without rescanning the document.

Each document is tagged with an id, thereby allowing us to easily identify data associated with a particular document. Every tuple representing a set of structured attributes has a document id associated with it. One problem typical with the WWW domain involves having different information on the same HTML page. In our model, this is represented as multiple tuples or concept sets within the same document. For example, on-line stores usually have information about multiple products on the same page. If each tuple represents a different product, then our model must capture the distinction between unique information associated with each product.

Suppose a winery maintains the following product descriptions on a single WWW page:

Wine 1 $150 Cabernet Sauvignon
Rich purple/ruby color with a dark cherry…

Wine 2 $125 Chardonnay
Full of flavor and fruit with moderate oak…

Our system should be able to distinguish wine 1 from wine 2. For this reason, we incorporate the notion of a subdocument. Each of the previous product descriptions is considered a subdocument. Similar to a document, every subdocument is represented with a unique id. A table in the database maps subdocuments to documents. Both the concepts in the concept library and the tuples in the database may be mapped to a subdocument instead of a document. Further, tuples are not duplicated in the database for a document and subdocument. The tuple is associated with the lowest subdocument level. An index that maintains a mapping between different document / subdocument pairs is the only additional overhead. This scheme allows us to distinguish information within a page, while still maintaining the flexibility to view information at the page level.

Another problem associated with HTML documents is that the document content can change over time. To account for this, we maintain a database field that contains an expiration date. The information discovery module ensures valid data in the database and the concept library.

Rule Generator

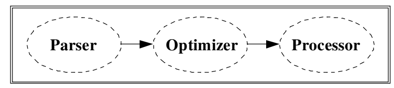

Figure 3: Rule Generator Components

At a high level, the rule generator runs a data mining algorithm given a set of user specified constraints. Figure 3 shows the components of the rule generator. The *parser* takes user specified information from the GUI and massages it for the optimizer. The *optimizer* looks at the type of rule the user wants to generate, the constraints specified, and determines which of its execution plans is most efficient. For example, suppose only structured values are specified by the user, but he wants to generate rules involving the specified value over an unconstrained concept domain. A strategy that identifies a common document set by initially retrieving the document lists associated with each structured value prior to retrieving the document lists associated with the entire concept space will be less costly than one that begins by retrieving the document lists associated with the concept space. This results because this execution reduces the candidate document space considerably. Finally, once the optimizer has determined the most efficient execution plan, the *processor* executes the plan and returns the results to the GUI. While executing the plan the processor verifies that the related documents have not expired. If they have, it sends a request to the information discovery module to update the information in the system for the expired documents. In this manner, a lazy update scheme of the database and concept library is employed.

Information Discovery Module

The information discovery module finds and parses documents. Figure 4 illustrates the components of the information discovery module. The key component of this module is the *extractor*. It populates the database and the concept library for every specialized domain. Each domain has a unique extraction program or set of programs that are knowledgeable about the organization of information in that domain. In other words, a domain expert gives specifications that can be used as a template for identifying structured values. An unsupervised domain specific extraction program can then use this knowledge to populate both the database and the concept library.

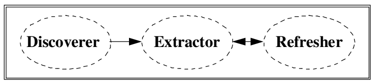

Figure 4: Information Discovery Module Components

The discoverer and the refresher are specifically included for more dynamic domains, e.g. WWW. The *discoverer* is an intelligent agent that looks at different search engines and tries to find pages associated with different specialized domains. If it finds something new, it informs the extractor, so that the document can be parsed. The extractor and the discoverer are both off-line processes that do not affect the performance of the rule generation algorithms. Further, because each document is only parsed once, the time needed to extract concept and tuple information is reasonable.

The *refresher* employs a dual update scheme to keep the data in the database and the concept library consistent. The first scheme is a proactive one that periodically checks the expiration date of the data and updates it if the expiration date is approaching. Since it is not always possible to keep everything up to date, the refresher may receive requests from the processor to update data in the database and the concept library.

User Interface

The user interface needs to be flexible enough to allow users to specify the following criteria: domain of interest, type of rule, attribute type, attribute value, concept value, confidence, support, and background knowledge.

The rule generator generates rules based on the input provided by the user. The user is not required to provide

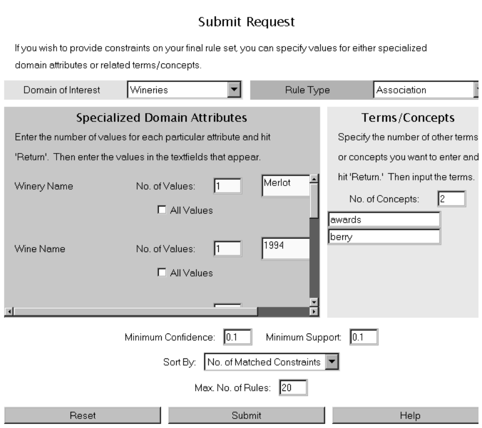

Figure 5: IRIS Input Interface

all of the criteria specified. For example, if he wants to generate rules using only structured values, no concepts need to be provided. If he wants to generate rules about a set of concepts, no structured values are necessary. In this manner, rules about any subset or superset of data components can be generated. Unconstrained or partially constrained classification and association rule generation can also be requested. If the user wants to generate rules involving some specified concepts over the entire domain of an attribute, he need only specify concepts of interest and request generation of a partially constrained rule set (i.e. a rule set only constrained by concepts).

Initial Prototype

We have developed IRIS (Inductive Rule Identification System), a prototype system that implements the architecture presented in section 3. For this prototype we are using an Oracle 7 DBMS to store our structured attribute values. We use linear hash tables to store the information in the concept library. The storage mirrors that of the extended concept hierarchy (ECH) as presented in (Singh, Scheuermann, and Chen 1997), where a separate ECH exists for each specialized concept guide. The initial prototype domain is winery WWW pages.

The list of concepts and relationships in the concept dictionary was developed using WordNet. WordNet, created at Princeton University, contains lexicographer files that organize nouns, verbs, adjectives, and adverbs into groups of synonyms and describes relations between synonym groups (Miller et al. 1990). All the terms that are not on our stoplist are maintained in the concept dictionary.

In contrast, the information in the specialized concept guide is extracted from the documents themselves. During the extraction process, we compare each meaningful concept to concepts in the concept dictionary. If the concept is in the concept dictionary, we initially assume it is a general concept. If the concept is not in the concept dictionary, it is placed in the specialized concept guide. As concepts are added, their specialized relationships to other concepts are determined using weighted frequency of co-occurrence measures (Salton and McGill 1983). Because this does not always result in semantically consistent relationships, we are investigating other approaches for generating specialized relationships, including synonym detection using natural language processing. Clearly if ontological information exists for a specialized domain, it should be incorporated into the specialized concept guide.

To populate the database and the specialized concept guide, we identified online wineries using WWW search engines. We then developed a template for extracting structured values. During the same pass of the document, our extraction tool also identified meaningful unstructured concepts. Using HTML tags and some natural language processing, the weight or importance of each concept within a document was determined. We then employ a

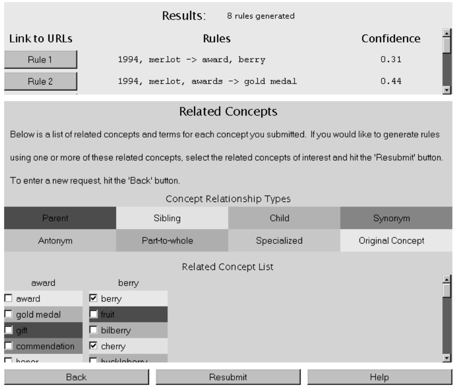

Figure 6: IRIS Output Interface

heuristic that uses this information along with frequency of occurrence information to identify relationships among domain concepts. Figure 5 illustrates our user input interface and some sample input values from the winery domain. This GUI is implemented using JAVA. Figure 6 shows the output interface with some sample output. In this example, rule 1 is a general rule that uses all the values specified by the user. Rule 2 is a child rule that incorporates knowledge about the child concept *gold medal*. For more details about rule types and the algorithm used to generate these rules, refer to (Singh, Scheuermann, and Chen 1997).

Conclusions

This paper proposes a system architecture for a semi-structured data mining tool. This architecture incorporates a compact representation of textual data that is stored in the concept library and the database. If both structures are not maintained, the final rule set is less efficient to generate. Without a database, numeric and categorical data is more difficult to handle. Without a concept library, the final rule set is not as semantically rich. We also believe there are a number of advantages to maintaining a distinction between the concept library and the database. First, it enables us to maintain unstructured concepts that are typically neglected in other representations of semi-structured data. Second, we can use the same concept library irrespective of the structure of the database. Third, we do not need to physically store the concept library and the database at the same site. As more and more domains are added to the text mining system, e.g. IRIS, it may be advantageous to store data on multiple servers. Moreover, separating the library allows us to easily add 'above' already existing databases. Along with the ability to accurately represent semi-structured data, our infrastructure distinguishes specialized domain concepts from general concepts, maintains a list of concept relationships, updates information that has expired, and generates different types of rules. Finally, the modularity of the architecture facilitates the addition of new domains. No restructuring is necessary. A new specialized concept guide is created to store the concept information and new tables are created to maintain the structured data values.

References

Agrawal R., Imielinski T., and Swami A. 1993. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data,* 207-216. Washington, DC: ACM Press.

Amir, A., Feldman, R., and Kashi, R. 1997. A New and Versatile Method for Association Generation. In *Proceeding of Principles of Data Mining and Knowledge Discovery, 221-231*. Trondheim, Norway: Springer.

Feldman, R. and Hirsh, H. 1996. Mining Associations in the Presence of Background Knowledge. In *Proceedings of the International Conference on Knowledge Discovery in Databases*, 343-346. Portland, Oregon: AAAI Press.

Lent, B., Agrawal, R., and Srikant, R. 1997. Discovering Trends in Text Databases. In *Proceedings of the International Conference on Knowledge Discovery in Databases.* Newport Beach, Calif.: AAAI Press.

Lagus, K., Honkela, T., Kashi, S., and Kohonen, T. 1996. Self-organizing Maps of Document Collections: A New Approach to Interactive Exploration. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 238-243. Portland, Oregon: AAAI Press.

Miller, G., Beckwith, R., Fellbaum, C., Gross, D. and Miller., K. 1990. Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography* 3(4): 235-244.

Singh, L. 1997. Automatic Preprocessing & Transformation of Semi-Structured Data for Data Mining Applications. Master's Thesis, Dept. of Electrical Engineering and Computer Science, Northwestern Univ.

Salton, G. and McGill, M. 1983. Introduction to Modern Information Retrieval. New York, New York: McGraw-Hill.

Singh, L., Scheuermann, P., Chen, B. 1997. Generating Association Rules from Semi-structured Documents Using an Extended Concept Hierarchy. In *Proceedings of the International Conference on Information & Knowledge Management*. Las Vegas, Nev.: ACM Press.

Tresch, M., Palmer, N., and Luniewski, A. 1995. Type Classification of Semi-structured Documents. In Proceedings of the International Conference on Very Large Data Bases, 263-274.Zurich, Switzerland: Morgan Kaufman.