An Algorithm for Constrained Association Rule Mining in Semi-structured Data

Lisa Singh[1], Bin Chen[1], Rebecca Haight[2], Peter Scheuermann[1]

[1] Northwestern University, Evanston, IL 60208, USA
{lsingh, bchen, peters @ ece.nwu.edu}
[2] Data Harbor, Inc., Chicago, IL 60610, USA
{rlh@dataharbor.com}

Abstract. The need for sophisticated analysis of textual documents is becoming more apparent as data is being placed on the Web and digital libraries are surfacing. This paper presents an algorithm for generating constrained association rules from textual documents. The user specifies a set of constraints, concepts and/or structured values. Our algorithm creates matrices and lists based on these prespecified constraints and uses them to generate large itemsets. Because these matrices are small and sparse, we are able to quickly generate higher order large itemsets. Further, since we maintain concept relationship information in a concept library, we can also generate rulesets involving concepts related to the initial set of constraints.

1 Introduction

Finding global patterns or generalizations can provide insight into subsets of a textual data set. However, because of the large number of patterns that typically exist in a textual database, preconstraining the data space and searching for targeted patterns can at times be more fruitful. This paper presents an algorithm for constrained association rule discovery. The main contributions of our algorithm are as follows. First, our algorithm creates matrices and lists based on prespecified user constraints and uses them to generate large itemsets. Because these matrices are sparse, we are able to quickly generate higher order large itemsets. Further, it incorporates knowledge from both the structured components and the unstructured components of the textual data set. Finally, since we maintain concept relationship information in a concept library, we can also generate rulesets involving concepts related to the initial set of constraints.

The remainder of this paper is organized as follows. Section 2 presents a motivating example from a business document collection. Section 3 describes our conceptual framework for generating rulesets from semi-structured data. Section 4 details our constrained association rule algorithm. Finally, Section 5 presents experimental results and conclusions.

2 Motivating Example

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

The problem with text data is that limited insight about documents can be attained using only the structured document components. A digital library system only provides limited knowledge about the document collection as a whole. For example, suppose a user is interested in answering the following question:

> *During the past year, how have articles in a particular journal been broken down in my research area?*

If the user is attempting to answer this question using a typical information retrieval system, he will enter the research area of interest (*organizational theory*) and a journal name (*Academy of Management*). When the list of articles is returned, he will need to find groupings of articles from the previous year and semi-automatically categorize them.

We propose an algorithm that not only answers this question, but also attempts to provide the user with some additional insight. We define a *constraint* to be any concept or structured value input by the user. For example, if the user specifies the following constraints (*1998, Academy of Management* and *organizational theory*), then our algorithm might return the following ruleset:

(A) Academy of Management, 1998 $\rightarrow$ organizational theory : 20%

(B) Academy of Management, organizational theory, 1998
$\rightarrow$ population ecology : 20%
$\rightarrow$ bureaucracy : 30%

Rule A involves the constraints specified by the user. This rule states that 20% of the articles in the *Academy of Management* in the *1998* journal are *organizational theory* articles. Rule B involves concepts related to the concept constraints (e.g. *organizational theory*). According to Rule B, in *1998*, *organizational theory* articles published in the *Academy of Management* journal were about *population ecology* 20% of the time and *bureaucracy* 30% of the time. We refer to these rules as *constrained association rules* since they are based on a set of prespecified constraints. In order for Rule B to be generated, we must maintain information about relationships that exist between the concepts, both the strength of the relationship and the type of relationship (e.g. synonym, broad-narrow, etc). By maintaining concept and structured value data, we can quickly extract patterns from a large document space.

3 Definitions and Conceptual Model

3.1 Association Rules

Formally, [1] defines an association rule in transactional databases to be an expression of the form $X \rightarrow Y$, where X and Y are sets of items or *itemsets*. The *support* of itemset XY is the probability of joint occurrence of X and Y, $P(XY)$. A *large itemset* is one in which $P(XY)$ is above the minimum support, *min_sup*. We will use large itemsets and *strong sets* interchangeably throughout the paper. The *confidence* of $X \rightarrow Y$ is defined as the conditional probability of Y given X, $P(Y|X)$, where *min_conf* represents the minimum confidence.

The definition of association rules in a transaction database can be extended to a semi-structured domain. Specifically, each document can be viewed as a transaction and each structured value or concept as an item. Even though we can model semi-structured data as a set of transactions, we choose to reformulate this model because text mining has different requirements than traditional transaction database mining. First, the number of distinct items is very large, and, therefore, interesting rules may have a very low support. Also, documents (transactions) are typically long and must be parsed offline to determine the meaningful set of concepts and structured values. Finally, online ontologies or dictionaries identify semantic relationships between concepts. This additional knowledge can be used to efficiently focus and add semantic knowledge to the final ruleset. Because of these distinctions, current structured value association rule algorithms that scan through all the items in all the transactions are not ideal for mining document data. Instead, it can be more advantageous to generate rules based on a set of constraints. For preconstrained textual mining, this structure facilitates the rule discovery process.

3.2 Conceptual Model

In our previous work [5], we proposed a system architecture that attempts to provide an infrastructure robust enough to facilitate the discovery of rules from semi-structured data sets. In this architecture, concepts are stored in the concept library, while structured values are stored in the database. Figure 1 shows some example structured value data and a mapping to the documents containing each structured value. We choose to associate each document name with a document id since storing varying length document names with each structured value uses more disk space. The concept library consists of two logical components. One is a mapping between concepts and documents containing each concept. Figure 2 shows examples of this. The other is a graph structure that maintains the relationships each concept has to other concepts in the domain, as well as the weight of each relationship. Figure 3 shows a portion of the concept library, where each concept, C_i, is a node and each relationship a weighted edge. Single directional edges point from broader to narrower concepts, while bidirectional arrows exist between similar sibling concepts. For every pair of concepts C_a and C_b, the *relationship weight* $rw(C_aC_b)$ identifies the strength of the relationship. There are

Structured Values (SV)	SV IDs	#. Docs	Document ID List
Joe Smith	S1	8	1, 3, 6, 8, 12, 16, 18, 20
... ...			
Administrative Science Quarterly	S5	6	1, 3, 8, 12, 15, 17
Academy of Management	S6	7	2, 4, 6, 9, 10, 16, 19
... ...			
1998	S8	10	1, 3, 4, 5, 6, 8, 10, 16, 18, 19

Fig. 1. Structured Value Document Mapping

different techniques for determining relationship weight. It can be manually determined by assigning a weight based on a defined relationship in a dictionary or ontology. Another approach uses information retrieval techniques to determine the weight based on the frequency of appearance of C_a and C_b in the same document. If $rw(C_aC_b)$ is greater than a minimum specified relationship weight, then we say that C_a and C_b have a *strong bond*. The notion of 'bonds' is derived from the bond-energy algorithm used in cluster analysis to derive similarities among attributes [7]. We use the bond to represent the relationship between two items. *Bond energy* or bond denotes the relationship weight. Document data can be

ConceptID	#. Docs	Document ID List
C1	10	1, 3, 4, 6, 7, 8, 11, 16, 18, 20
C2	2	2, 5
C3	8	1, 3, 6, 8, 11, 16, 18, 20
C4	8	5, 7, 8, 10, 12, 15, 16, 18
C5	4	2, 9, 13, 19
C6	3	5, 6, 10
C7	4	2, 3, 8, 10
C8	6	5, 10, 12, 15, 17, 19

Fig. 2. Concept Document Mapping

stored as traditional transaction systems, where the document id corresponds to the transaction id and the concepts or structured values map to transaction items. Our model differs from this traditional approach. Instead of storing each document transaction in the database, we choose to associate transaction ids with each item. In other words, for a given item (concept or structured value), we maintain a mapping to transaction (document) ids in the concept library and the database. Specifically, the problem of discovering large itemsets reduces to

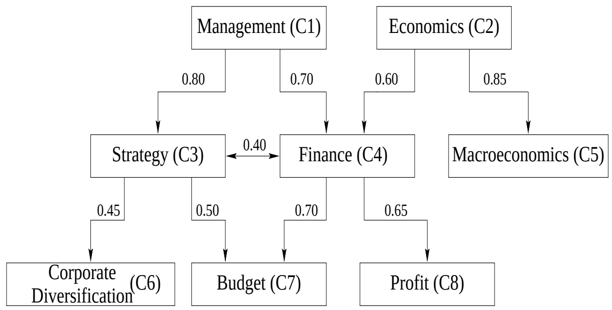

Fig. 3. Concept Library Data

the problem of traversing sparse matrices.

A small amount of literature exists about semi-structured text mining algorithms [2, 3, 6]. Of those, [6] is the only algorithm that uses prespecified constraints in it. The other works propose unsupervised procedures. As previously mentioned, for semi-structured data sets, unconstrained mining is very slow and leads to large numbers of rulesets.

Our previous work in [6] also identified associations among concepts and a single structured value. The user specifies one concept C_1, one structured value S_1, a minimum support min_sup, and a minimum confidence min_conf. The algorithm begins by obtaining the document ids of each constraint. The support for the potential large itemset $P(C_1 S_1)$ is calculated. If it is above min_sup, we search for large itemsets containing *concept_relatives* C_r of constraint C_1. For each concept C_r, if $rw(C_1 C_r)$ is larger than a minimum specified relationship weight, $P(C_1 S_1 C_r)$ is calculated. Rules are generated for all $P(C_1 S_1 C_r)$ above min_sup. Rules above min_conf are returned to the user.

To extend this algorithm to allow users to specify multiple constraints, it is necessary to introduce new data structures so as to avoid the computation of $P(C_i S_j C_r)$ for all combinations of concept constraints, C_i, structured value constraints, S_j, and *concept_relative*, C_r.

4 Constrained Association Rule Algorithm

We have extended our previous algorithm in the following ways:

1. We allow users to specify an unlimited number of constraints.
2. We use two sparse matrices to help us avoid comparing document lists for every subset of structured data values and concepts.
3. When determining large itemsets involving *concept_relatives*, we do not need to calculate the probability of joint occurrence for every *concept_relative* and the current large itemset. Instead we eliminate *concept_relatives* that clearly cannot be a member of a new large itemset.

4. Our concept library stores not only parent-child-sibling relationships, but others as well, synonym, part-to-whole and is-a. This in turn implies that a semantically more accurate set of rules can be generated. It should be noted that our concept library is a compilation of multiple extended concept hierarchies (ECHs) as proposed in [6].

During the development of our proposed algorithm, we will make use of the user inputs in Figure 4. Figure 5 highlights the main features of our algorithm.

STRUCTURED VALUES	**CONCEPTS**
AUTHOR:	strategy
Joe Smith	finance
PUBLICATION NAME:	management
Administrative Science Quarterly	profit
Academy of Management	
PUBLICATION DATE:	
1998	
MINIMUM SUPPORT:	0.1%
MINIMUM CONFIDENCE:	60%
MINIMUM RELATIONSHIP WEIGHT:	30%

Fig. 4. User Inputs

The algorithm begins by obtaining the document ids of all the structured values and all the concepts originally requested by the user, where N_c and N_s are the number of concepts and structured values specified by the user, respectively. The structured value information is obtained from the database, while the concept information is obtained from the concept library. For our example query illustrated in Figure 4, $N_s = 4$ and $N_c = 4$.

For all the specified constraints, C_i and S_j, Step 2 verifies that they are large 1-itemsets by checking the count of the document lists. The concept constraints above min_sup are placed in the *C-List*, a list containing all the large concept itemsets. Similarly, the structured value constraints above min_sup, 0.1% for our example, are placed in the *SV-List*, a list containing all the large structured value itemsets.

In Step 3, we generate a structured value matrix,*SV-Bond Matrix*, that will be used to determine higher order large itemsets. In order to create the matrix, we use the items in the SV-List to generate large 2-itemsets. We determine if $P(S_a S_b)$ is above min_sup for all pairs (a, b) in the list. If so, the pair (a, b) is added to the SV-List and a 1 is added to the SV-Bond Matrix to indicate a strong bond between two structured values. If pair (a, b) is less than min_sup a 0 is added.

Notice that we only populate the lower triangle of the matrix. At this stage, the sum of each column is calculated. This *pair count* identifies the number of

Algorithm: Generalized Constrained Association Rules

Input: C_i - concept constraints, (i: 1 to N_c)
S_j - structured value constraints, (j: 1 to N_s)
Output: RS - ruleset containing constraints and related concepts

1. Get document ids for all C_i and S_j
2A. for every C_i
    if $support(C_i) >$ min_sup add to C-List
2B. for every S_j
    if $support(S_j) >$ min_sup, add to SV-List
3. Create structured value matrix, SV-Bond Matrix
4. Determine structured value large item sets, put in SV-List
5. Create concept matrix, C-Bond Matrix
6. Determine concept large itemsets, put in C-List
7. Identify large itemsets using lists
8. for each large itemset, determine $concept\_relatives$
    generate additional large itemsets using $concept\_relatives$
9. Calculate RS

Fig. 5. Pseudo-code for Constrained Association Rule Alg.

strong bonds existing in the column. Figure 6 shows the SV-Bond Matrix for our example. Column S_1 has the largest pair count. Just based on these counts alone, we know that the maximum large itemset size cannot exceed 3 and that it must include column S_1.

Structured Value IDs (columns) × Structured Value IDs (rows)

	S1	S5	S6	S8
S1	0	0	0	0
S5	1	0	0	0
S6	0	0	0	0
S8	1	1	1	0
Pair Count	2	1	1	0

Fig. 6. Example SV-Bond Matrix

Concept IDs (columns) × Concept IDs (rows)

	C1	C3	C4	C8
C1	0	0	0	0
C3	1	0	0	0
C4	1	1	0	0
C8	0	0	1	0
Pair Count	2	1	1	0

Fig. 7. Example C-Bond Matrix

The matrix can now be used to generate higher order large itemsets (Step 4 of algorithm) without first determining the intersections of every subset of structured values. Since all subsets of a large itemset must have strong bonds, we can use the information in the SV-Bond Matrix to find these larger sets. There are three criteria for determining the large itemsets of size N, where N is larger than 2:

1. The pair count for one of the N columns containing the data values is at least N - 1.
2. Each pair in the set must have a strong bond.
3. The set must have support above the min_sup.

We determine large 3-itemsets by using the strong bonds in the SV-Bond Matrix. For each set of size 2, we check the pair count for the first item in the set. If it is at least 2, we test whether it can be expanded by adding a structured value that has a strong bond to all of the elements in the current set. This new 3-itemset is a *potential large itemset*. Once all the potential large itemset of size 3 have been found, the joint probability of each potential large itemset is calculated. Those sets with a joint probability above min_sup are added to the SV-List. This continues until either no new set is found or the pair count for all the columns in the matrix is less than N - 1. Usually, the SV-Bond Matrix is not dense. Therefore, much work is saved by not testing every combination of the input set.

Based on the SV-Bond Matrix in Figure 6, $(S_1 S_5 S_8)$ is the only potential large 3-itemset. Notice that it is unnecessary to check for any potential large 3-itemsets beginning with S_5 since the pair count for the column is 1. If we assume the document collection contains 3000 documents, then $P(S_1 S_5 S_8) = 3 \div 3000 = 0.1\%$ and $(S_1 S_5 S_8)$ is a large itemset. Because no column has a pair count of 3, we know that there are no large 4-itemsets.

A similar matrix called the *C-Bond Matrix* is used to identify large itemsets involving concepts (Step 5 of algorithm). However, the process is less costly because intersections of document id lists is avoided when creating the matrix. This results because we have relationship information in the concept library.

In order to create the matrix, we determine if the relationship weight between every pair of concepts in the C-List is above the minimum specified relationship weight. If so, a 1 is added to the C-Bond Matrix. Figure 7 shows the C-Bond Matrix generated from our example data assuming the minimum relationship weight is 0.30.

Once the C-Bond Matrix is generated, Step 6 finds higher order large itemsets using the same approach as that presented in Step 4. One difference is that the potential large itemsets do not need to be verified. Each potential large itemset is an actual large itemset. Recall that in Step 4 the supports of all the potential large itemsets needed to be verified at every level by intersecting the document lists.

This is avoided because relationship weight overrides support. It is a more general weight that holds across individual databases within a particular domain. Therefore, if every pair of concepts in a set has a strong relationship, the set is a large itemset. Because we will be generating *mixed rulesets* that contain both structured values and concepts, we perform a union of the document lists associated with the large concept itemset. Therefore, the support for the large concept itemset is 1. In this manner, we ensure that the concept component of any mixed ruleset is large in and of itself. All concept large itemsets are added to the C-List.

Based on the C-Bond Matrix in Figure 7, $(C_1C_3C_4)$ is the only large 3-itemset. It should be noted, that in order to have a large N-itemset containing concepts, all the concepts in the large itemset must form a fully connected subgraph. Returning to Figure 3, we see that C_1, C_3 and C_4 is a fully connected subgraph.

At this stage in the algorithm, we have two lists, one consisting of structured value strong itemsets and the other of concept strong itemsets. These lists are used to generate mixed large itemsets. We calculate the support by intersecting the document lists of the structured value large itemset and the concept large itemset and keep the potential large itemset if it is above min_sup. This large itemset can now be used in rules. For example, the following large itemsets can be generated: $C_3C_4S_1S_5S_8$.

To generate large itemsets including related concepts or themes not prespecified by the user, *concept_relatives* with strong bonds must be determined (Step 8 of algorithm). The purpose of generating these itemsets is to facilitate the discovery of new concept sets that might otherwise have been overlooked by the user. We generate sets involving *concept_relatives* by taking the relatives of one concept in a large mixed set and checking whether or not all the other concepts in the large itemset are related to the *concept_relative*. If all the concepts in the large itemset are related to the *concept_relative*, a new large itemset composed of the current large itemset and the relative is created. This step is repeated for each mixed large itemset.

For our example, suppose we are attempting to find the *concepts_relatives* of the large itemset $C_3C_4S_5S_8$. Concepts C_6 and C_7 are *concept_relatives* of concept C_3, but only C_7 is also a relative of C_4. Therefore, the large itemset $C_3C_4C_7S_5S_8$ is created.

The final step involves generating the actual rules. Similar to other works, we specify rules by finding antecedents and consequents above min_conf. An additional step we include involves checking the relationship type information to attempt to identify a rule with a meaningful semantic relationship. One of the rules generated from the large itemset $C_3C_4C_7S_5S_8$ is the following: 67% : C_3, C_4, S_5, $S_8 \rightarrow C_7$. Strategy and finance related articles written in Administrative Science Quarterly in 1998 focus on budgets 67% of the time.

5 Results

5.1 Data Set

Our data set consists of over 50,000 documents from the ABI/Inform Information Retrieval System. ABI/Inform maintains bibliographic information, abstracts, and documents from over 800 business related journals. We only use a small subset of the documents in the ABI/Inform IR System. The structured data values associated with each article are stored in an Oracle 7 relational database on an HP 700 series workstation over an NFS file system. Examples of structured value tables include author, publication, and location.

5.2 Experiments

During our experiments, both the C-Bond Matrix and the SV-Bond matrix were not very full: on average the S-Bond Matrix was 7% full and the C-Bond Matrix 10% full. We were interested in determining if a bound existed on the average number of relationships a concept has. Therefore, we sampled concepts in WordNet, a general online ontology [4] and found that concepts had an average of 25 relationships. This implies that most columns in the C-Bond Matrix are sparse.

Likewise, the sparse nature of the SV-Bond Matrix is also a result of a textual data set. For example, multiple publication values cannot ever occur in the same document. Further, a typical author publishes with a small number of co-authors (150 in a lifetime) in a small number of journals (100 or so). Since a user will typically enter a mix of all different types of structured values, a relationship bound will exist.

Our experimental results for execution time of our algorithm averaged over 10 runs are shown in Figures 8, 9, 10. Figures 8 and 9 show that the overall time is quasi-linear with respect to the number of concepts entered and the number of structured values entered, respectively. For Figure 8, the number of structured values was held constant at 5. Similarly, in Figure 9, the number of concepts was also held constant at 5. Figure 10 shows the total running time with respect to the total number of

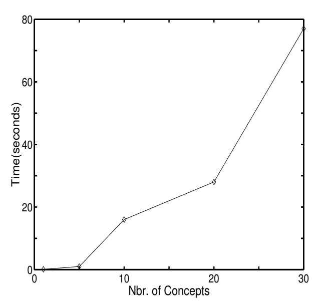

Fig. 8. Time vs. Nbr. of concepts

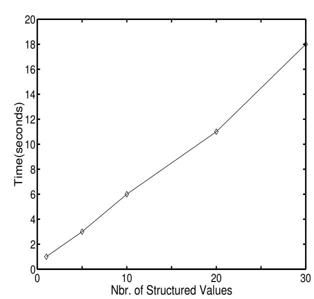

Fig. 9. Time vs. Nbr. of structured values

5.3 Conclusions

We have introduced an algorithm for generating constrained association rules. It incorporates knowledge from both the structured and unstructured components

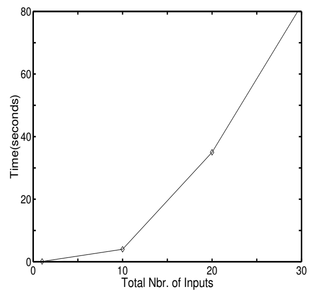

Fig. 10. Total running time

of the data set and generates rulesets using concepts related to the original set of constraints. It also uses some simple data structures that reduce the overall complexity of the algorithm.

This algorithm has only been tested on one semi-structured data set. We hope to continue our testing on other semi-structured data sets. At this stage we have not compared this algorithm against those proposed for structured transaction databases. We need to determine the sparsity of structured data sets to evaluate whether or not our approach is a good option in that domain. Finally, the rulesets presented here are fully constrained rulesets. We are also interested in partially constraining the document space and evaluating performance under those circumstances.

References

1. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," In *Proceedings of the International Conference of Very Large Databases*, September 1994.
2. A. Amir, R. Feldman, R. Kashi, "A new and versatile method for association generation," In *Information Systems*, Vol 22, 1997.
3. R. Feldman and H. Hirsh, "Mining associations in the presence of background knowledge. In *Proceedings of the International Conference on Knowledge Discovery in Databases*, 1996.
4. G. Miller. WORDNET: An on-line lexical database. *International Journal of Lexography*, 1990.
5. L. Singh, B. Chen, R. Haight, P. Scheuermann, and K. Aoki, "A robust system architecture for mining semi-structured data," In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 1998.
6. L. Singh, P. Scheuermann, and B. Chen. "Generating association rules from semi-structured documents using an extended concept hierarchy," In *Proceedings of the International Conference on Information and Knowledge Management*, November 1997.
7. T. Teorey and J. Fry. *Design of Database Structures*, Prentice Hall, 1982.