Proceedings of the CIKM-2002 (short paper),
pp.681-683, McLean, VA, November 2002

Mining Soft-Matching Association Rules

Un Yong Nahm   Raymond J. Mooney
Department of Computer Sciences, TAY 2.124
The University of Texas at Austin
Austin, TX 78712-1188
{pebronia,mooney}@cs.utexas.edu

ABSTRACT

Variation and noise in database entries can prevent data mining algorithms, such as association rule mining, from discovering important regularities. In particular, textual fields can exhibit variation due to typographical errors, mispellings, abbreviations, etc.. By allowing partial or "soft matching" of items based on a similarity metric such as edit-distance or cosine similarity, additional important patterns can be detected. This paper introduces an algorithm, SOFTAPRIORI that discovers soft-matching association rules given a user-supplied similarity metric for each field. Experimental results on several "noisy" datasets extracted from text demonstrate that SOFTAPRIORI discovers additional relationships that more accurately reflect regularities in the data.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems—*Textual Databases*; H.2.8 [**Database Management**]: Database Applications—*Data Mining*; I.2.7 [**Natural Language Processing**]: Text Analysis

General Terms

Algorithms, Performance, Experimentation

Keywords

Text Mining, Association Rules, Textual Databases, Noisy Databases

1. INTRODUCTION

Textual entries in many database fields exhibit minor variations that can prevent mining algorithms from discovering important regularities. Variations can arise from typographical errors, misspellings, abbreviations, as well as other sources. Variations are particularly pronounced in data that is automatically extracted from unstructured or semi-structured documents or web pages. One approach to this problem is to standardize the name of each entity using either a manual "data cleaning" process or an automated "de-duping" procedure. In that case, however, discovered associations are not able to capture all similarities between different items.

In this paper, we explore the alternative of directly mining "dirty" data by discovering "soft matching" association rules whose antecedents and consequents are evaluated based on sufficient similarity to database entries. Similarity of text can be measured using standard "bag of words" metrics [4] or edit-distance measures. We generalize the standard APRIORI algorithm for discovering association rules [1] to allow for soft matching based on a given similarity metric for each field. We present experimental results on several datasets demonstrating that SOFTAPRIORI discovers additional relationships that more accurately reflect regularities in the data.

2. BACKGROUND AND RELATED WORKS

The problem of mining association rule is to discover all association rules that have support and confidence greater than the user-specified minimum support and minimum confidence. One of the popular algorithms for discovering association rules is APRIORI [1] where the closure property of itemset support was introduced. Association rule mining has been applied directly to textual data; however, the heterogeneity of items in textual databases has often been overlooked. Compared to data cleaning methods that impose a single normalization on the data items, mining soft-matching rules dynamically clusters data items into different groups depending on the association under consideration.

3. MINING SOFT ASSOCIATION RULES

In this section, we introduce the problem of mining *soft* association rules from databases and investigate how to utilize an existing association rule mining algorithm to incorporate similarity in discovering associations.

3.1 Soft Association Rules

We define soft relations as follows, assuming that a function, $similarity(x, y)$, is given for measuring the similarity between two items x and y. The range of the similarity function is the set of real numbers between 0 to 1 inclusive.

DEFINITION 1 (IS-SIMILAR-TO). *An item x is similar to an item y ($x \sim y$) iff $similarity(x, y) \geq T$, where T is a predefined threshold between 0 and 1. We also define a binary function $similar(x, y)$ which is 1 if $x \sim y$ and 0 otherwise.*

DEFINITION 2 (IS-A-SOFT-ELEMENT-OF). *An item x is a soft-element of an itemset I ($x \in_{soft} I$) iff there exists an $x' \in I$ such that $x' \sim x$.*

DEFINITION 3 (IS-A-SOFT-SUBSET-OF, SET-SIMILAR). *An itemset I is a soft-subset of an itemset J ($I \subseteq_{soft} J$) iff for every item in I there is a distinct similar item in J. Two sets I and J are similar, denoted by $I \sim J$, iff $I \subseteq_{soft} J$ and $J \subseteq_{soft} I$. I is a proper soft-subset of J iff $I \subseteq_{soft} J$ holds but $I \sim J$ is not true.*

The following is a formal statement of the problem of mining soft association rules: Let $I = \{i_1, i_2, ..., i_m\}$ be a set of literals,

```
Input: D is the set of records.
Output: L_k is the frequent k-itemsets.
Function SoftApriori (D)
L_1 := FindFrequentItemsets(D).
k := 2
while (L_{k-1} ≠ ∅) do
    C_k := GenerateCandidates(L_{k-1})
    forall records r ∈ D do
        forall c ∈ C_k do
            if c ⊆_soft r
            then c.count := c.count + 1
    L_k := All candidates in C_k with minimum softsups
    k := k + 1
Return ⋃_k L_k.
```

Figure 1: The SOFTAPRIORI algorithm

called items. Let $\mathcal{D}$ be a set of records, where each record R is a set of items such that $R \subseteq I$. A soft association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$ and no item in X is a soft-element of Y. The problem of mining soft association rules is to find all soft association rules, $X \Rightarrow Y$, such that the *soft-support* and the *soft-confidence* of $X \Rightarrow Y$ are greater than the user-defined minimum values. Formal definition for soft-support is straightforward generalization of the traditional one as given below.

DEFINITION 4 (SOFT-SUPPORT). *The soft-support of an itemset X in a set of records (database) $\mathcal{D}$, denoted as $softsup(X)$, is the number of records, $R \in \mathcal{D}$, such that $X \subseteq_{soft} R$. The soft-support of a rule $X \Rightarrow Y$ in a database $\mathcal{D}$, denoted as $softsup(X \Rightarrow Y)$, is the number of records $R \in \mathcal{D}$ such that $X \cup Y \subseteq_{soft} R$.*

3.2 The SoftApriori Algorithm

To discover frequent itemsets for soft association rules, we generalize the existing itemset mining algorithm presented in [1] in a straightforward way. Since the notion of equality in the traditional definition of an association rule is replaced by similarity, we need to compute the soft-support of each item and itemset by Definition 4. In this approach, frequent itemsets under the definition of soft-support (Definition 4) are treated as normal items and the standard APRIORI algorithm can be used with minor modifications.

Figure 1 gives pseudocode for the SOFTAPRIORI algorithm. Notations such as L_k (set of frequent k-itemsets) and C_k (set of candidate k-itemsets) are from [1]. The first step of the algorithm determines the frequent 1-itemsets. We assume the minimum soft-support value, $minsup$, is provided by the user. The set of frequent 1-itemsets L_1 in SOFTAPRIORI is defined as follows:

$$L_1 = \{\{x\} \mid x \in I \wedge softsup_I(x) \geq minsup\}$$

By Definition 4, the soft-support of each item is calculated by summing the number of occurrences of all similar items. Formally, the soft support of a 1-itemset $\{x\}$ where x is an element of the set of all items I ($x \in I$) is computed as follows:

$$softsup_I(\{x\}) = \sum_{y \in I} similar(x, y) \times support(y)$$

While counting the occurrences of all items, we measure the similarity of every pair of items and construct an $m \times m$ matrix $similar(i, j)$, where m is the total number of items in the database. To determine frequent 1-itemsets, the soft-supports of all items are computed. Intuitively, we construct a cluster of items containing the items similar to each given "central" item, and sum the support of all items in the cluster. After constructing a set of frequent items, they are treated the same as items in the original APRIORI algorithm. In a manner similar to the initial construction of frequent items, itemsets are grown by computing the soft-support of candidates and discarding those with low soft-support.

The extra complexity of constructing a similarity matrix is $O(m^2)$ where m is the total number of items since we need to compute the similarity of every pair of items. However, by treating every pair of items in different fields as non-similar, we can lower the number of similarity computations to $\sum_{k=1}^{N} {m_k}^2$ whereas N is the number of fields and m_k is the number of items in field k.

3.3 Implementation I: String Edit-Distance

We implemented SOFTAPRIORI by modifying a publicly available version of APRIORI [2]. We defined $similarity(x, y)$ based on normalized edit-distance to measure similarity of string-valued items. Given a particular edit-distance function, we can reduce the time complexity of determining similar items. Since edit distance counts the number of operations needed to change one string to another, two strings cannot be similar if their lengths are too different. We can reduce the number of comparisons between items even further by using an n-gram index. For any given string x, one can retrieve a list of strings worth comparing by determining the minimum number of n-grams of x that must be shared with any similar string y. In our implementation, we used a trigram index to efficiently retrieve a list of candidate similar strings for each string.

3.4 Implementation II: Vector Space Model

In this implementation, we adopted the "bag-of-words" model for representing each item and measured the similarity between items by the cosine similarity from information retrieval [4]. In the current version of our system, users can plug any similarity metric into each field out of the four options: edit-distance, cosine similarity, absolute difference for numeric values, and absolute difference converted the dates. In terms of computational performance, the major bottleneck is the worst-case quadratic time complexity of measuring the similarity of many pairs of items. Fortunately, there are well-known indexing methods that allow efficient identification of items that are close in cosine similarity [4]. In our implementation, we used an inverted index to retrieve similar items efficiently.

4. EXPERIMENTAL EVALUATION

In this section, we evaluate SOFTAPRIORI on three "dirty" databases extracted from text and compare prediction accuracies (measured on independent test data) of soft association rules and hard association rules mined from the same training data.

4.1 Datasets and Sample Rules

For the first dataset, 600 computer-science job postings to the newsgroup `austin.jobs` were collected to construct a textual database of job requirements. A second dataset was built from 300 computer-science resumes from the newsgroup `misc.job.resumes`. Finally, 3,000 science fiction (SF) book descriptions are automatically extracted from the `Amazon.com` online bookstore. The job postings dataset has 1,362 total items while the resume has 4,283 items and the book has 17,341 items. Examples of interesting soft association rules mined are shown in Figure 2. With the same values of confidence and support, SOFTAPRIORI discovers more general rules including frequent clusters of similar items that would be overlooked by the traditional algorithm because of the low support values for individual items. By combining Implementation I and II, we are able to specify string edit-distance as the similarity metric for shorter strings and cosine similarity for longer fields.

4.2 Experimental Methodology

We measured the ability of both hard and soft association rules mined from the same training data with the same minimum con-

Job postings (600)

1. database (databases, database sys.) $\in$ area $\Rightarrow$ oracle (oracle7) $\in$ application [3.2%, 43.2%]
2. mfc $\in$ area $\Rightarrow$ windows (windows nt, windows 95, windows 3.1, windows 3.x, windowsnt, windows95, windows'95) $\in$ platform [2.7%, 39.0%]

SF books (3,000)

1. $\{$mike(1), resnick(1)$\}$ $\in$ author **and** $\Rightarrow$ $\{$american(1), fantasy(1), fiction(4), science(3)$\}$ $\in$ subject [0.2%, 35.0%]
2. $\{$asimov(1), janet(1)$\}$ $\in$ author **and** $\{$fiction(2), robots(1), science(1)$\}$ $\in$ subject $\Rightarrow$ $\{$asimov(1), isaac(1)$\}$ $\in$ author [0.1%, 100.0%]

Figure 2: Sample soft association rules (T = 0.7)

Domain	Rule	Precision	Recall	F-Measure
Job	Soft	89.44	8.68	15.82
	Hard	86.92	8.55	15.57
Resume	Soft	89.45	3.13	6.06
	Hard	69.75	1.92	3.73
Books	Soft	88.47	10.55	19.06
	Hard	66.67	0.32	0.63

Table 1: Test accuracies of soft vs. hard association rules (%)

fidence and support parameters to make accurate predictions on the same disjoint set of test data. To determine the accuracy of a set of association rules, we measured precision and recall with respect to predicting the presence of items in a record from other items in that record. A prediction is judged to be correct iff there is an item in the record that is at least similar to the predicted item (i.e. $similarity(x, y) \geq T$). Antecedents of hard rules are matched using the appropriate hard matching criteria and soft rules are matched using the appropriate soft-matching criteria; however, predictions are always judged "softly" in order not to give soft rules an unfair advantage. Detailed pseudocode for the evaluation method is presented in [3].

4.3 Results and Discussion

The experimental results obtained for the four textual databases with the Implementation I are summarized in Table 1. This table gives average prediction accuracies for hard and soft association rules using a minimum support and confidence of 10% and 70% respectively for USENET postings and 2% and 70% for book descriptions, and using a similarity threshold of 0.7 for every field. Minimum support for book data is lower since otherwise no rules at all are found from this data. The results show that the accuracy of soft rules is consistently, significantly higher than that of hard rules. Training accuracy, measured by training and testing on the same entire dataset, shows similar patterns. We also performed the same experiments while varying these parameters [3]. Overall, the results clearly show that soft rules are generally better than hard rules at discovering reliable regularities in "dirty" data.

4.4 Performance Results

Finally, we present results showing the efficiency gained by using the optimization methods presented in Section 3.3 for quickly finding similar string-valued items. The 3,000 book descriptions in SF were used in this experiment with items in short-strings fields. Figure 3 shows the CPU time for each item in the similarity computing step. The "No Optimization" method stands for comparing all pairs of items in each field, "String Length" uses a heuristic to eliminate comparisons between strings with very different lengths,

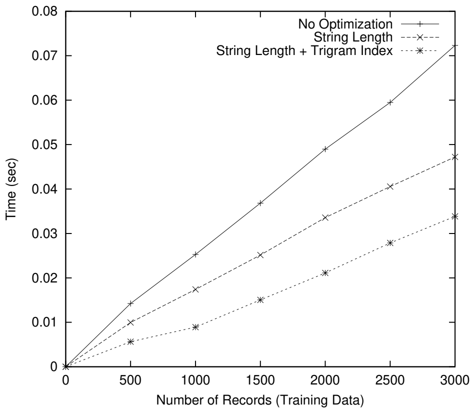

Figure 3: Running time for similarity computations (Implementation I)

and "String Length + Trigram Index" additionally employs the trigram index to retrieve strings with shared trigrams. Similar results from Implementation II with the cosine similarity as the similarity metric are presented in [3].

5. CONCLUSION AND FUTURE WORK

Data mining methods generally require terms in discovered rules to exactly match database entries. Normal variation in data items can therefore prevent the discovery of important and interesting relationships. We presented an algorithm which discovers "soft matching" rules that are evaluated using a specified similarity metric. Experimental results in several domains illustrate that soft-matching allows discovery of additional interesting rules that more accurately capture certain relationships. Allowing the discovery of soft-matching rules can eliminate the need for certain types of tedious data cleaning prior to knowledge discovery.

The limitation of the current definitions for soft-support and soft-confidence is that they do not reflect the different *original support* values of individual items nor different degrees of similarities between items. One possible solution to this problem is to redefine the similarity matrix as $similarity(i, j)$ instead of the binary value, $similar(i, j)$. Since the similarity of two textual items can vary depending on the specific domain, automatic learning or dynamic setting of threshold values should also be explored.

Acknowledgements

This research was supported by the National Science Foundation under grant IIS-0117308.

6. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB-94)*, pages 487–499, Santiago, Chile, Sept. 1994.

[2] C. Borgelt. Apriori version 2.6. http://fuzzy.cs.UniMagdeburg.de/~borgelt/, 2000.

[3] U. Y. Nahm, M. Bilenko, and R. J. Mooney. Two approaches to handling noisy variation in text mining. In *Papers from the Nineteenth International Conference on Machine Learning (ICML-2002) Workshop on Text Learning*, pages 18–27, Sydney, Australia, July 2002.

[4] G. Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989.