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, misspellings, abbreviations, etc. By allowing partial or “soft matching” of items based on a similarity metric such as edit-distance, 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
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 [11, 20]. For example, in data on local job offerings that we automatically extracted from news group postings, the Windows operating system is variously referred to as “Microsoft Windows”, “MS Windows”, “Windows 95/98/ME”, etc.

<table>
<thead>
<tr>
<th>ID</th>
<th>Areas</th>
<th>Platforms</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data Management</td>
<td>Windows</td>
<td>Microsoft Access</td>
</tr>
<tr>
<td>2</td>
<td>DB Management</td>
<td>Windows98</td>
<td>MS Access</td>
</tr>
<tr>
<td>3</td>
<td>Web</td>
<td>WindowsNT</td>
<td>VBScript</td>
</tr>
<tr>
<td>4</td>
<td>WW Web</td>
<td>WinNT</td>
<td>ASP</td>
</tr>
<tr>
<td>5</td>
<td>Network</td>
<td>Novell</td>
<td>Firewall</td>
</tr>
</tbody>
</table>

Table 1: Sample noisy textual database

For instance, consider the example in Table 1, that lists required skills for a set of computer-science jobs. In this database, the co-occurrence of “DB Management (or Data Management)” in areas, “Windows (or Windows98)” in platforms, and “Microsoft Access (or MS Access)” in applications is a pattern that a human can easily recognize (from jobs 1 and 2). However, traditional association rule mining techniques cannot discover such patterns because they treat “Data Management” / “DB Management”, “Windows” / “Windows98”, and “Microsoft Access” / “MS Access” as different items.

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 (a.k.a. “record linkage,” “merge/purge”, and “database hardening” [6, 17]). In that case, however, discovered associations are not able to capture all similarities between different items. For example, assume that we cluster “Windows”, “Windows98”, and “WindowsNT” and treat them all as an identical term, “Windows”; then we lose another association such as “Web (or WW Web)” and “WindowsNT (or WinNT)” often occurring together (in jobs 3 and 4). The dilemma of the fixed clustering strategy is that “WindowsNT” partially matches “WinNT”, but it also matches “Windows”.

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 [24] or edit-distance measures [25]; other standard similarity metrics can be used for numerical and other data types. 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: MINING ASSOCIATION RULES

The following is a formal statement of the problem of mining association rules: Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let $D$ be a set of records (the database), where each record $R$ is a set of items such that $R \subseteq I$. We say that a record $R$ contains $X$, a set of items in $I$ when $X \subseteq R$. An association rule is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ has support $s$ in the record set $D$ if $s$ is the number of records in $D$ that contain $X \cup Y$. The rule $X \Rightarrow Y$ holds in the record set $D$ with confidence $c$ if $c \%$ of the records in $D$ that contain $X$ also contain $Y$. Given a database $D$, the problem of mining association rules is to discover all association rules that have support and confidence greater than the user-specified minimum support and minimum confidence.

The classical application of the association rule mining techniques is market basket analysis about finding associations between items purchased by customers. An association rule from a supermarket database, “beer -> pretzels [20%, 80%]” indicates that 20% (support) of customers bought beer and pretzels together and 80% (confidence) of those who bought beer also bought pretzels.

One of the popular algorithms for discovering association rules is APRIORI [1] where the closure property of itemset support was introduced. APRIORI is based on breadth-first search and therefore ensures that the support values of all subsets of a candidate are known in advance. For each stage $k$, all candidates of a cardinality $k$ are counted in a scan over the database. APRIORI prunes all candidate itemsets such that any subset of that itemset is not frequent. A hash-tree data structure is used to store the candidates in each record for efficient look-up.

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. With a softened definition for associations that do not require exact matches, we are able to mine more general rules. We present an algorithm called SOFTAPRIORI for discovering soft association rules, as well as an implementation using a string edit-distance as the primary similarity metric.

3.1 Soft Association Rules

Before presenting our algorithm for discovering soft association rules, we define soft relations as follows. We assume that a function, $\text{similarity}(x, y)$, 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, and $\text{similarity}(x, y) = 1$ if $x = y$.

**Definition 1 (IS-SIMILAR-TO).** An item $x$ is similar to an item $y (x \approx y)$ if $\text{similarity}(x, y) \geq T$, where $T$ is a predefined threshold between 0 and 1. We also define a binary function similar($x, y$) which is 1 if $x \approx y$ and 0 otherwise. This definition is a natural generalization of ‘$x$ equals to $y (x = y)$’ with $T$ set to 1. The similarity relation is reflexive ($x \approx x$) and symmetric ($x \approx y$ implies $y \approx x$), but not transitive ($x \approx y$ and $y \approx z$ does not necessarily imply $x \approx z$). An item $x$ is not similar to an item $y (x \not\approx y)$ if $x \approx y$ does not hold.

<table>
<thead>
<tr>
<th>Record</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a, b, c, d$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$a', b', c', d'$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$a, c, c', d'$</td>
</tr>
<tr>
<td>$R_4$</td>
<td>$a', d, e$</td>
</tr>
</tbody>
</table>

**Table 2:** An example of a database with soft-matching items

**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 $x' \approx x$.

**Definition 3 (IS-A-SOFT-SUBSET-OF, SET-SIMILAR).** An itemset $I$ is a soft-subset of an itemset $J (I \subset_{soft} J)$ iff for every item in $I$ there is a distinct similar item in $J$, i.e. for every item $x_i \in I, I = \{x_1, \ldots, x_m\}$, there is an item $y_j \in J$ such that $x_i \approx y_j$ and $y_j \neq y_{j' \neq i}, 1 \leq j \leq m$. Two sets $I$ and $J$ are similar, denoted by $I \approx J$, iff $I \subset_{soft} J$ and $J \subset_{soft} I$. $I$ is a proper soft-subset of $J$ iff $I \subset_{soft} J$ holds but $I \approx J$ is not true.

**Definition 4 (SOFT-DISJOINT).** Two itemsets $I$ and $J$ are soft-disjoint when no item in $I$ is a soft-element of $J$. The “soft-disjoint” relation is symmetric, i.e. if $I$ and $J$ are soft-disjoint then so are $J$ and $I$.

For instance, consider an example in Table 2. Let us assume that those items with ‘$’ are similar to items with the similar literal without ‘$’, but not similar to those with other literals. With this assumption, $a$ is similar to $a' (a \approx a')$ and $a'' (a \approx a'')$, but not similar to $b (a \not\approx b)$. $R_1 \subset_{soft} R_2 \subset_{soft} R_3$ since every item in $R_1$ is similar to some item in $R_2$. $R_2$ is also a soft-subset of $R_3$ and that makes $R_1$ and $R_2$ similar to each other. However, $R_3 \not\subset_{soft} R_2$ since $c$ and $c'$ in $R_3$ have only one shared similar item $c$ in $R_2$, but a one-to-one mapping is required for soft-matching items.

The following is a formal statement of the problem of mining soft association rules: Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let $D$ 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 $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$. 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 (called minsup and minconf respectively). Formal definitions for soft-support and soft-confidence, which are straightforward generalizations of the traditional ones, are given below.

**Definition 5 (SOFT-SUPPORT).** The soft-support of an itemset $X$ is a set of records (database) $D$, denoted as $\text{softsup}(X)$, is the number of records, $R \in D$, such that $X \subseteq_{soft} R$. The soft-support of a rule $X \Rightarrow Y$ in a database $D$, denoted as $\text{softsup}(X \Rightarrow Y)$, is the number of records $R \in D$ such that $X \cup Y \subseteq_{soft} R$.

**Definition 6 (SOFT-CONFIDENCE).** The soft-confidence of a rule $X \Rightarrow Y$, denoted as $\text{softconf}(X \Rightarrow Y)$ is given by:

$$\text{softconf}(X \Rightarrow Y) = \frac{\text{softsup}(X \Rightarrow Y)}{\text{softsup}(X)}$$
For example, the soft-support of the itemset \( \{a, c\} \) for the database shown in Table 2 is 3 since it is a soft-subset of 3 records, \( R_1 \), \( R_2 \), and \( R_3 \). The soft-confidence of the association rule, \( \{a, c\} \Rightarrow \{b'\} \) is computed by dividing the soft-support of \( \{a, c, b'\} = (\{a, c\} \cup \{b'\}) \) by the soft-support of \( \{a, c\} \). Since the soft-support of \( \{a, c, b'\} \) is 2 (\( R_1 \) and \( R_2 \)), the soft-confidence of this rule is \( 2/3 \), or 66.67%.

### 3.2 The SoftApriori Algorithm

The problem of discovering soft association rules can be decomposed into three parts as in traditional association rule mining [1, 26]:

1. **Discover Frequent Itemsets:** Find all itemsets for which soft-support is greater than the user-specified minimum soft-support.

2. **Rule Generation:** Use the discovered frequent itemsets to generate the association rules that have higher soft-confidence than the user-specified minimum soft-confidence.

3. **(Optional) Rule Filtering:** Prune uninteresting rules from this set.

In this paper, we concern the first part, finding all frequent itemsets with higher soft-support than the user-specified minimum. Given the frequent itemsets, the Apriori algorithm [1] can be used to generate rules by simply replacing the confidence measure with soft-confidence.

In the current algorithm, we also add an extra constraint to the definition of similar items that avoids a practical problem. In most applications of mining association rules from textual databases, we do not expect similar items to appear together in the same database record. In other words, even though a single record contains string-valued items that are similar by Definition 1, such items generally refer to different entities. For instance, “ASP” and “JSP” are best considered distinct items despite their similar appearance when they both occur in the job skills of a single resume. Based on this intuition, the definition of similar items in the context of a given database is restated as follows.

**Definition 7 (iS-Similar-To (in-The-Context-Of)).** An item \( x \) is similar to an item \( y \) in the context of database \( D \) (\( x \sim y \)) iff similarity\((x, y) \geq T \) (where the threshold \( T \) is a predefined constant between 0 and 1) and \( x \) and \( y \) do not appear together in any record \( R \) in \( D \) (\( \{x, y\} \subseteq R \in D \)).

For brevity, in the rest of the paper, we use the shorter notation \( \sim \) without specifying the database \( D \) when the relevant database is clear from context.

### 3.2.1 An Overview of the 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 5. Similarity between items is computed once and cached for future references. In this approach, frequent itemsets under the definition of soft-support (Definition 5) are treated as normal items and the standard Apriori algorithm can be used with minor modifications.

**Input:** \( D \) is the set of records.

**Output:** \( L_k \) is the frequent \( k \)-itemsets.

**Function SoftApriori \((D)\)**

\[
L_1 := \text{FindFrequentItemsets}(D), \quad k := 2
\]

while \( (L_{k-1} \neq \emptyset) \) do

begin

\[
C_k := \text{GenerateCandidates}(L_{k-1})
\]

for all records \( r \in D \) do

for all \( c \in C_k \) do

if \( c \sim r \) then

\( c \cdot \text{count} := c \cdot \text{count} + 1 \)

end

\( L_k := \text{All candidates in } C_k \text{ with minimum softsup} \)

\( k := k + 1 \)

end Return \( \bigcup_k L_k \).

Figure 1: The SoftApriori algorithm

<table>
<thead>
<tr>
<th>Record</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( a, b, c, d' )</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( e, j, g )</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>( a, c, h, i )</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>( j, d, i, k )</td>
</tr>
</tbody>
</table>

Table 3: Sample database

Figure 1 gives pseudo code 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].

### 3.2.2 Finding Frequent Itemsets

The first step of the algorithm determines the frequent 1-itemsets. We assume the minimum soft-support value, \( \text{minsup} \), is provided by the user. The set of frequent 1-itemsets \( L_1 \) in SoftApriori is defined as follows:

\[
L_1 = \{\{x\} \mid x \in I \land \text{softsup}(x) \geq \text{minsup}\}
\]

In other words, \( L_1 \) is the set of all 1-itemsets whose soft-support is greater than the user-specified minimum. By Definition 5, 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:

\[
\text{softsup}(\{x\}) = \sum_{y \in I} \text{similarity}(x, y) \times \text{support}(y)
\]

While counting the occurrences of all items, we measure the similarity of every pair of items and construct an \( m \times m \) matrix \( \text{similarity}(i, j) \), where \( m \) is the total number of items in the database. Usually the similarity matrix is extremely sparse since most items are not similar. A hash table is used to store the sparse similarity matrix. Given the database in Table 3, the similarity matrix is shown in Table 4.

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. The similarity hash table is used to efficiently retrieve similar items. For example, assume \( I = \{\text{powerpoint, MSpowerpoint, Macintosh, Mac, Mac Powerpoint}\} \). The soft-support of powerpoint is the sum of the support of powerpoint itself and those of MSpowerpoint and Mac Powerpoint, which
3.2.3 Generating Candidate Itemsets and Computing Soft-Support

After constructing a set of frequent items, they are treated the same as items in the original APRIORI algorithm. Note that the closure property on which the original APRIORI algorithm is based still holds for soft itemsets. In other words, if an itemset has a soft-support higher than minsup, then every subset of that itemset also has soft-support higher than minsup.

Given $L_{k-1}$, the set of all frequent $(k - 1)$-itemsets, the candidate itemsets $C_k$ are generated by self-joining $L_{k-1}$ with $L_{k-1}$. We also follow the pruning strategy presented in [1] by deleting all itemsets $c \in C_k$ such that some $(k - 1)$-subset of $c$ is not an element of $L_{k-1}$.

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 soft-subset function is used to check which itemsets in $C_k$ are softly in record $r$. For each itemset that is a soft subset of $r$, or for each set of items that have similar items in $r$, soft-support of an $k$-itemset is again computed by the equation in Definition 5, counting the number of soft-matching items, instead of simply counting the number of occurrences of each item.

3.2.4 Time Complexity of the Algorithm

The extra complexity of constructing a similarity matrix in the initial stage 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, this complexity can be reduced in practice because items in different fields do not need to be compared. By treating every pair of items in different fields as non-similar, we are able to 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 $k$.

Depending on the particular similarity metric, additional optimizations are possible. For example, items in numeric fields can be sorted and then similar items can be quickly determined by checking neighboring items in order of proximity until the similarity threshold is exceeded. We present an additional optimization for string edit-distance in Section 3.3.2. In Section 6, we discuss additional approaches that would allow finding all similar pairs of items in $O(m)$ time.

3.3 Implementation

We implemented SOFTAPRIORI by modifying a publicly available version of APRIORI [3]. In accordance with this implementation, we considered only the case of a single item in the consequent of an association rule.

3.3.1 Similarity Metrics: String Edit-Distance

To measure similarity of string-valued items, a form of edit-distance was adopted. Edit-distance is defined as the minimum number of primitive edit operations, such as character addition, substitution, and deletion, required to transform one string to another. However, edit-distance cannot be used directly since it returns 0 if the strings are identical and greater values when they are different. Therefore, we defined $\text{similarity}(x, y) = 1 - \text{normalized edit distance}(x, y)$ where normalized edit-distance is scaled to always be between 0 and 1 (based on the lengths of the two strings).

In our implementation, we used affine gap cost [22, 17], an edit distance originally developed for gene/protein sequence comparison. Affine gap cost incurs one penalty for starting a new gap (i.e. sequence of deletions) and a typically smaller penalty for continuing an existing gap (i.e. contiguous deletions). The match cost, mismatch cost, gap start cost, and gap extend cost, which are parameters of affine gap cost, are set to 0, 3, 3, and 1, respectively. We have also tested a more standard edit-distance, Levenshtein distance [14], but found the affine gap cost provides more intuitive results. For example, “Austin” and “Houston” are considered to be similar with a threshold of 0.7 by Levenshtein distance, but not similar by affine gap cost. White spaces contained in strings are considered as a blank character and upper and lower cases are not distinguished.

The computation of edit-distance is performed by dynamic programming in time $O(nm)$ when $n$ and $m$ are the lengths of the two input strings. However, the edit-distance computation in our implementation does not always require the full $O(nm)$ time because it stops as soon as the intermediate result exceeds the minimum value computed from the given similarity threshold, $T$.

The normalized edit-distance of affine gap cost for two
strings is defined as follows:

\[
\text{normalized edit distance}(x, y) = \frac{\text{affine gap cost}(x, y)}{\text{strlen}(x) + \text{strlen}(y) + 4}
\]

(1)

where \(\text{strlen}(x)\) returns the length of \(x\).

The similarity of items in numeric fields is defined as the normalized absolute difference, computed as follows:

\[
similarity_{np}(x, y) = 1 - \frac{|x - y|}{\max(D) - \min(D)}
\]

(2)

where \(\max(D)\) and \(\min(D)\) are the maximum and the minimum values for that field in the given database \(D\).

3.3.2 Optimization

Given a particular edit-distance function, we can reduce the time complexity of determining similar items (see Section 3.2.4). 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. For example, we do not have to compute the actual affine gap cost for "windows-nt" and "unix" when \(T = 0.7\) to confirm they are different because the gap between any 10-character string and any 4-character string is too big to result in a similarity greater than 0.7. By generalizing this observation, we obtain the following formula for determining if two strings can not be similar under affine gap cost:

\[
\frac{|\text{strlen}(x) - \text{strlen}(y)| + 2}{\text{strlen}(x) + \text{strlen}(y) + 4} \leq 1 - T
\]

(3)

Using this test, we are able to eliminate edit-distance computations for very different strings.

We can reduce the number of comparisons between items even further by using an n-gram index [12]. An n-gram is a substring of length \(n\) of a given string. It is easy to show that a string \(x\) cannot be similar to \(y\) for any reasonably high threshold \(T\), if they do not share a common substring. In addition, a string \(x\) cannot be similar to \(y\) for a given threshold if they do not share at least \(k\) characters of \(n\)-grams. For example, two similar strings "opera" and "opera 6" (affine gap cost = 4) share 3 n-grams, e.g. "opera", "per", and "era". 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. Each string is indexed under every three-character substring that it contains. For example, "opera" is indexed by "ope", "per" and "era" while "opera 6" is indexed by "ope", "per", "era", "ra", and "a 6". To find candidates for similar strings, we use the index to efficiently retrieve all strings that share at least one trigram. It can be shown that no string \(x\) can be similar to another \(y\) under affine gap cost with \(T \geq 0.7\), unless they share at least one trigram. Without loss of generality, assume that \(|x| \leq |y|\) where \(|x| \geq 1\) and \(|y| \geq 1\) are the numbers of characters in \(x\) and \(y\). To be similar to each other, the normalized edit-distance between \(x\) and \(y\) must be less than \(1 - T\). For any \(x\), there is no \(y\) that does not share a trigram with \(x\) that makes the normalized affine gap cost edit-distance of \(x\) and \(y\) less than 0.3 by Equation 1.

In general, the minimum number of n-grams that need to be shared by \(x\) and \(y\) to make them similar can be represented by a function of \(|x|\), \(|y|\), \(n\) (the length of n-grams) and the cost parameters for the affine gap cost. Our implementation uses \(\frac{|y|}{10}\) as the minimum number of n-grams that must be shared by similar strings. In other words, a string is compared only to strings that share at least \(\frac{|x|}{10}\) n-grams when \(T = 0.7\). It can be shown that when \(T = 0.7\), \(\frac{|y|}{10}\) is always greater than the minimum number of n-grams that need to be shared to make \(x\) and \(y\) similar. Therefore, this method is guaranteed not to miss any pair of similar items. Experiments that demonstrate the efficiency gained by this approach are presented in Section 4.4.

4. EXPERIMENTAL EVALUATION

Our focus on mining soft-matching rules was originally motivated by text-mining research on discovering patterns in data automatically extracted from natural-language documents and web pages [19, 20, 21]. Therefore, in this section, we evaluate SOFT A PRIORI 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 and information on programming languages, platforms, applications, areas, company, recruiter, job title, required years of experience, desired years of experience, salary, post date, city, state and country were identified to construct a textual database of job requirements. A second dataset was built by extracting similar information from 300 computer-science resumes from the newsgroup misc.job.resumes. Both of these datasets were originally constructed for training automatic information-extraction systems for these domains [4]. The current experiments utilize only the original human-tagged data in order to avoid introducing errors in the evaluation of test accuracy caused by extraction errors in automatically extracted data.

Finally, 3,000 science fiction (SF) book descriptions and another 3,000 science book descriptions automatically extracted from the Amazon.com online bookstore are used. The information extractor (wrapper) for Amazon was developed manually and is highly accurate. This dataset was originally built for a book recommending system [18]. 10 fields (title, author, type, publisher, publication date, subjects, related books, related authors, price, and average rating) are used in our experiments.

The job postings dataset has 600 records with 1,362 total items, the resume dataset has 300 records with 4,283 items, SF books has 3,000 records with 17,341 items, and science books has 3,000 records with 5,587 items (22,775 items for all books).

The values of some quantitative fields, such as "required-years-of-experience" in job postings and "publication-date" in book descriptions, were converted to numeric data using a simple preprocessor. Dates were converted to numbers using the transformations: convert(Date) = (Year – 1990) × 12 + Month where Year and Month are extracted from Date.

Examples of interesting soft association rules mined from the four sets of data are shown in Figure 2 and Figure 3. Items similar to a given item are shown in parentheses, and
values for softup and softconf are shown in brackets. For example, the first rule says if "TX" is in the state field and "US" is in country then either "Austin" or "Austin" (which is a common misspelling for "Austin") is in city. Although a similar hard association rule, \texttt{tx} $\in$ state and \texttt{us} $\in$ country $\Rightarrow$ \texttt{austin} in city can be discovered using traditional association rule mining, it lacks useful information about similar items. With the same values of confidence and support, \textsc{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.

Other examples show that soft association rules are able to capture patterns based on groups of similar items such as the different versions of Windows, Fortran, etc. Rules from the book descriptions show that \textsc{softapriori} is also able to mine rules with similar numeric values, e.g. average-rating. In the book data, it frequently discovered rules including types or non-standardized variations, such as "Isaac Asimov", "Isaac Aminov", and "Isaac Asimov" or "Arthur C. Clarke" and "Arthur Charles Clarke" in the author field.

We presented items in soft association rules by specifying the item itself followed by its corresponding similar items in parentheses. The issue of presenting rules in a user-friendly manner depends on the representation of textual items and the similarity metric used. For other similarity metrics, alternative approaches to representing "softness," such as ranges on numeric values, may be appropriate. If textual items are treated as bags-of-words, as in cosine similarity, representing an item by the bag intersection of its similar items in the cluster might be a suitable choice [21].

### 4.2 Experimental Methodology

The goal of data mining is to discover accurate and interesting knowledge from data. To avoid overly-optimistic estimates, accuracy of discovered knowledge should always be measured on independent test data. Consequently, we measured the ability of both hard and soft association rules mined from the same training data with the same minimum confidence and support parameters to make accurate predictions on the same disjoint set of test data. To obtain statistically reliable estimates of accuracy, we employed ten-fold cross-validation which averages performance over 10 trials of training on 90% of the data and testing on 10% [16].

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. Precision is the percentage of predicted items that are actually present and recall is the percentage of actual items that are correctly predicted. We also report $F$-measure which is the harmonic mean of recall and precision:

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

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. $\text{similarity}(x, y) \geq T$). Antecedents of hard rules are matched using the appropriate hard matching criteria and soft rules are matched using the appropriate soft-matching
Input: $D_{test}$ is the test database.

Rules is the association rule set.

Output: Precision and Recall as measured on $D_{test}$.

Function ComputeAccuracy ($D_{test}$, Rules)

num_fired := num_matched := 0

foreach record $R \in D_{test}$ do

/* precision */

foreach rule $(A \Rightarrow c) \in Rules$ do

if $(\{rule\} \text{ is hard and } A \subseteq R) \text{ or } (\{rule\} \text{ is soft and } A \subseteq_{\text{soft}} R)$

then if rule is hard then $A' := A$

else $A' := X \text{ s.t. } X \subseteq R$ and $X \sim A$

num_fired := num_fired + 1

if $c \sim r$ and $(\{rule\} \text{ is hard and } A \subseteq R - \{r\}) \text{ or } (\{rule\} \text{ is soft and } A \subseteq_{\text{soft}} R - \{r\})$

then num_predicted := num_predicted + 1

/* recall */

foreach $r \in R$ do

num_item := num_item + 1

if there exists a rule $(A \Rightarrow c) \in Rules$ s.t.

$c \sim r$ and $(\{rule\} \text{ is hard and } A \subseteq R - \{r\}) \text{ or } (\{rule\} \text{ is soft and } A \subseteq_{\text{soft}} R - \{r\})$

then num_predicted := num_predicted + 1

Precision := num_matched / num_fired

Recall := num_predicted / num_item

Return (Precision, Recall).

Figure 4: Method for evaluating precision/recall

<table>
<thead>
<tr>
<th>Domain</th>
<th>Rule</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job</td>
<td>Soft</td>
<td>89.44</td>
<td>8.65</td>
<td>15.52</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>86.92</td>
<td>8.55</td>
<td>15.57</td>
</tr>
<tr>
<td>Resume</td>
<td>Soft</td>
<td>89.44</td>
<td>3.13</td>
<td>6.06</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>69.75</td>
<td>1.92</td>
<td>3.73</td>
</tr>
</tbody>
</table>

Table 6: Test accuracies of soft vs. hard association rules on USENET postings (%)

<table>
<thead>
<tr>
<th>Domain</th>
<th>Rule</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF</td>
<td>Soft</td>
<td>88.47</td>
<td>10.55</td>
<td>19.06</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>66.67</td>
<td>0.32</td>
<td>0.63</td>
</tr>
<tr>
<td>Science</td>
<td>Soft</td>
<td>72.74</td>
<td>26.30</td>
<td>35.71</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SF + Science</td>
<td>Soft</td>
<td>90.88</td>
<td>5.22</td>
<td>9.87</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>81.25</td>
<td>0.34</td>
<td>0.68</td>
</tr>
</tbody>
</table>

Table 7: Test accuracies of soft vs. hard association rules on book descriptions (%)

<table>
<thead>
<tr>
<th>Minconf (%)</th>
<th>Rule</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Soft</td>
<td>90.30</td>
<td>84.27</td>
<td>83.72</td>
<td>79.94</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>57.13</td>
<td>54.06</td>
<td>52.40</td>
<td>59.19</td>
</tr>
<tr>
<td>50</td>
<td>Soft</td>
<td>90.86</td>
<td>86.35</td>
<td>86.36</td>
<td>86.76</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>62.19</td>
<td>60.41</td>
<td>60.32</td>
<td>60.27</td>
</tr>
<tr>
<td>60</td>
<td>Soft</td>
<td>90.79</td>
<td>87.71</td>
<td>86.45</td>
<td>83.46</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>66.64</td>
<td>64.47</td>
<td>62.16</td>
<td>64.80</td>
</tr>
<tr>
<td>70</td>
<td>Soft</td>
<td>91.34</td>
<td>89.45</td>
<td>85.76</td>
<td>83.93</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>71.51</td>
<td>69.75</td>
<td>74.50</td>
<td>76.92</td>
</tr>
<tr>
<td>80</td>
<td>Soft</td>
<td>92.14</td>
<td>88.37</td>
<td>84.13</td>
<td>81.39</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>78.84</td>
<td>79.05</td>
<td>80.60</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Table 8: Precision of soft vs. hard rules (Resume)

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. For example, the training accuracies of soft rules on the job postings domain are 89.62% precision, 8.68% recall, and 15.83% F-measure, while the results for hard rules are 86.82%, 8.65%, and 15.74%. The larger differences in accuracy between soft and hard rules for the resume data compared to the job postings data can be explained by the more homogeneous nature of the job data. The job data were collected from one local newsgroup during a relatively short period (a couple of months). On the other hand, the resume data is more diverse since they were gathered from a general newsgroup archive that includes postings up to two years old. As an illustration, the job postings data has an average of 1.69 programming-language items per record, compared to 4.55 for the resume data.

Although the similarity threshold is set at 70% throughout the previous experiments, we also tested other values such as 60% or 80%. The results were similar, e.g. average precision/recall for job postings (10% minconf, 70% minconf) with $T = 0.6$ are 89.15%/9.37% (soft rules) and 86.92%/8.65 (hard rules). With $T = 0.8$ and the same minconf and minconf, the average precision and recall for soft rules on the resume data are 92.42% and 2.71% while those for hard rules are 86.32% and 1.92%, respectively.

To see if soft association rules were still more accurate for other combinations of minimum confidence and support, we performed the same experiments while varying these parameters. For resume data, Table 8 shows precision and Table 9 shows recall for various values of minimum confidence and support. No hard rules are found for 20% support and 80% confidence. 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 3.3.2 for quickly finding similar string-valued items. The 6,000 book descriptions in SF and science were used in this ex-
### Table 9: Recall of soft vs. hard rules (Resume)

<table>
<thead>
<tr>
<th>Minconf (%)</th>
<th>Rule</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hard</td>
<td>3.07</td>
<td>2.97</td>
<td>2.91</td>
<td>2.88</td>
</tr>
<tr>
<td>60</td>
<td>Soft</td>
<td>3.17</td>
<td>3.14</td>
<td>3.13</td>
<td>2.49</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>3.01</td>
<td>2.76</td>
<td>2.31</td>
<td>2.07</td>
</tr>
<tr>
<td>80</td>
<td>Soft</td>
<td>3.15</td>
<td>3.11</td>
<td>2.82</td>
<td>2.82</td>
</tr>
<tr>
<td></td>
<td>Hard</td>
<td>2.25</td>
<td>1.46</td>
<td>0.69</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Figure 5: Running time for similarity computations

The experiment after removing items in non-string fields such as publication date, price, and average rating. The total number of items for this dataset is 20,197 and the average string length is 25.16 characters.

Figure 5 shows the CPU time for the similarity computation step. Differences between every pair of performance results were statistically significant (p < 0.05). 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, and “String Length + Trigram Index” additionally employs the trigram index to retrieve strings with shared trigrams. This experiment was performed on a Linux/i686 P.C. The results demonstrate that with good heuristics and an efficient indexing method, our approach is scalable to larger datasets by reducing the total number of explicit similarity comparisons between pairs of items.

### 5. RELATED WORK

Association rule mining has been applied directly to textual data [8, 11]; however, to our knowledge, the heterogeneity of items in textural databases has not been addressed. In [21], an inductive learning method was introduced that integrates rule-based and instance-based methods for mining soft-matching rules from textual data. Compared to inducing rules sufficient for prediction, SOFT APRIORI finds all association rules with a given soft-support and soft-confidence, and therefore typically discovers a larger set of regularities, which is preferable in certain data-mining tasks.

Compared to automated data cleaning methods that impose a single normalization on the data items [6, 17], mining soft-matching rules dynamically clusters data items into different groups depending on the association under consideration, i.e., each discovered rule may group items into different similarity-based equivalence classes. For example, “Windows NT” must be placed either in the “NT" or “Windows" group in the normalization approach, while our algorithm allows it to belong to both clusters.

Mining association rules over interval data [15] is similar to our work in that it also first tries to find meaningful clusters and discovers associations from interval clusters. However, it concerns continuous or ordered data such as numeric values while we focus on non-transitive similarity relations between discrete items. Similarly, ARCS [13] only considers continuous spaces in binning the data and clustering association rules mined from the binned data.

Generalized association rules [26] were introduced to discover interesting rules when an item taxonomy is available. An item hierarchy provides pre-determined item clusters at multiple levels of abstraction; however, it must be provided by the user in advance and does not address the issue of unpredictable variation and noise in the items. Another modification of APRIORI is presented in [27] that allows random noise or error in the data; however, it does not address the issue of soft-matching items based on similarity.

WHIRL is a query processing system that combines traditional database and information retrieval methods by introducing a “soft join” operation [5]. WHIRL and SOFT APRIORI share a focus on soft-matching rules; however, WHIRL is not a data mining system and does not discover such rules.

### 6. FUTURE WORK

Our current implementation focuses on textual data in which items are short strings for which edit-distance is a suitable similarity metric. An obvious extension is to replace edit-distance with “bag of words” similarity metrics such as vector-space cosine similarity from information retrieval [24].

Another extension is to incorporate semantic information in the similarity metric. One approach is to use lexical knowledge-bases such as WordNet [9] to determine a measure of semantic similarity between words, such as that used in [2]. Another approach is to utilize statistical measures of semantic similarity based on an analysis of word co-occurrences, such as that used in Latent Semantic Indexing (LSI) [7]. Other sorts of user-defined knowledge could also be incorporated into similarity metrics. For example, domain-specific information such as that “Dallas” is geographically close to “Austin” but not to “San Francisco” could be used in measuring similarity of items in a city field.

In terms of computational performance, the major bottleneck is the worst-case quadratic time complexity of measuring the similarity of all pairs of items. We presented methods that dramatically reduce the number of comparisons by utilizing the division of items into different fields, sorting numerical items, and employing information on lengths and n-grams for string-valued items. Efficient methods for retrieving similar items for additional similarity metrics are needed. Fortunately, there are well-known indexing meth-
ods that allow efficient identification of items that are close in edit-distance [12], Euclidean distance [10], or cosine similarity [24, 5]. It is desirable to have new interestingness measures for specifying how strongly a rule is supported or how tightly items are associated. 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 or different degrees of similarities between items. One possible solution to this problem is to redefine the similarity matrix as \( \text{similarity}(i, j) \) instead of the binary value, \( \text{similar}(i, j) \). Soft-support of an item could be redefined accordingly, e.g. as the sum of similarities between that item and all soft-matching items in the database.

Currently the threshold values for determining similarity are pre-determined and fixed throughout the mining phase. Since the similarity of two textual items can vary depending on the specific domain, automatic learning or dynamic setting of threshold values should be explored. Automatic learning of parameterized similarity metrics based on labeled training examples of similar and dissimilar items [23] could also be exploited.

7. CONCLUSION

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. The SOFTAPRIORI algorithm 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.

Acknowledgements

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

8. REFERENCES


