Discovering Correlations in Annotated Databases

A Major Qualifying Project

Submitted to the Faculty of

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree in Bachelor of Science

In

Computer Science

By

__________________________________

Stephen Donohue

May 5, 2015

____________________________________

Professor Mohamed Eltabakh, Project Advisor

This report represents the work of WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its website without editorial or peer review. For more information about the projects program at WPI, please see http://www.wpi.edu/academics/ugradstudies/project-learning.html
ABSTRACT

Annotated databases contain large amounts of metadata information from which new knowledge can be derived. The goal of this project was to implement means of discovering and managing the association rules between annotations and data, with special focus given to the incremental maintenance of these association rules when new annotations are added to the database. After successfully implementing this system, the association rules were used to predict missing annotations for data records.
# TABLE OF CONTENTS

Abstract 2

1. Introduction 4
2. Related Work 6
3. Preliminaries 8
4. Discovery of Correlations 12
5. Exploitation of Correlations 24
6. Conclusions and Future Work 26
7. References 27
1. INTRODUCTION

1.1 Overview

In modern relational database systems there are various types of metadata information, commonly referred to as *annotations*. These annotations consist of information such as versioning timestamps, execution statistics, related comments or articles, corrections and conflict-related information, and auxiliary exchanged knowledge from different users. Figure 1 presents an example of an annotated database. The black pins reference an article related to the data record while the red flag annotation indicates the information is incorrect.

![Figure 1: Example of Annotation-Related Correlation](image)

From this metadata, these annotations, valuable information can be gathered through the discovery of correlations that exist in annotated databases. This project focuses on two types of correlations: those between data records and annotations (data-to-annotation) and those between annotations and other annotations (annotation-to-annotation). As the number of annotations increases, often outnumbering the actual data records, it becomes increasingly important to efficiently process them and utilize them in the data processing cycle.
This paper will describe the effort to create an efficient system for not only the discovery of association rules in annotated databases, but the dynamic maintenance of these rules as the databases are modified. We will first introduce the ideas and work that were essential to the development of this project. Then, we will discuss the various approaches we took to accomplishing our goal, as well as what we learned from them.

The goal of this project was to create a system responsible for the management of annotations in relational databases. A user needed to be able to manage annotations within a database with as little manual curation as possible. Toward this end the system needed to include an association rule miner to discover data-to-annotation and annotation-to-annotation correlations, update and discover new association rules upon the addition of new annotations or data records, and suggest the addition or removal of annotations to data records based upon the discovered association rules.

1.2 Focus and Goals

- Discover generalization-based correlations
- Create a system for the incremental maintenance of correlations when adding new annotations or tuples.
- Exploit the discovered correlations to enhance the quality of the annotated database.
2. RELATED WORK

2.1 Annotation Management in Relational Databases

Annotation management techniques are meant to enable users to attach extra information to data records in databases. An annotation may be attached to cells, rows, columns, or arbitrary sets and combinations of such. There exist some systems which provide GUI, such as the one shown in Figure 2, through which annotations may be added, and others which extend SQL with new commands and clauses to enable the addition of annotation.

Figure 2: Excel Based GUI for Annotation Management
2.2 Association Rules Mining

Association rule mining is the process of discovering correlations within large datasets. Association rules are presented in the following form:

\[ X \Rightarrow Y, \text{ support } = \alpha, \text{ confidence } = \beta \]

This means that the presence of the left hand side (L.H.S.) itemset \( X \) implies the presence of the right hand side (R.H.S.) itemset \( Y \) with support equal to \( \alpha \) and confidence equal to \( \beta \). The support is computed as a fraction of transactions (tuples) containing \( X \cup Y \) relative to the size of the database.

Confidence is computed as \( \text{support}(X \cup Y) / \text{support}(X) \). Therefore given a minimum support and minimum confidence, the association rule mining technique should discover all rules have support and confidence above the specified values.

A related extension to the standard association rule mining problem is the mining of multi-level rules. In this extension, the technique is given a domain generalization hierarchy over one or more attributes, and we need to discover the association rules that may span different levels of the hierarchy. For example, in market analysis, the items “pants”, “shirts”, and “t-shirts” can be generalized to “clothes”. Because of this generalization, some rules may hold at the higher level(s) of the hierarchy which may not be true for the lower more-detailed levels.
3. PRELIMINARIES

3.1 Discovering Association Rules with the Apriori Algorithm

Apriori is an algorithm used for frequent itemset mining and association rule learning over transactional databases.[4] This algorithm served as a starting point for the project, as a means for first discovering association rules across our database, and for verifying the validity of later work such as the incremental maintenance of association rules.

```
Apriori(T, ε)
    L₁ ← {large 1 - itemsets}
    k ← 2
    while Lₖ₋₁ ≠ emptyset
        Cₖ ← {a ∪ {b} | a ∈ Lₖ₋₁ ∧ b ∈ \( \bigcup Lₖ₋₁ \) ∧ b \∉ a}
        for transactions t ∈ T
            Cₜ ← {c | c ∈ Cₖ ∧ c \⊆ t}
        for candidates c ∈ Cₜ
            count[c] ← count[c] + 1
        Lₖ ← {c | c ∈ Cₖ ∧ count[c] ≥ ε}
        k ← k + 1
    return \( \bigcup Lₖ \)
```

**Figure 3: Apriori Algorithm**

Figure 3 presents the apriori algorithm. The algorithm uses breadth-first search and a hash tree structure to count candidate item sets. It generates itemsets of length k from itemsets of length k-1. It then prunes the itemsets which are infrequent.

The only modification made to the algorithm for this project was to introduce the early elimination of any candidate patterns that didn’t include at least one annotation.
Figure 4: Example of Dataset

The dataset that was used throughout this project is shown in Figure 4. Each line represents one tuple within the dataset. For example, the first line represents a tuple with annotations indicated by IDs “Annot_4” and “Annot_5.” The numbers shown are IDs for the actual values within the tuples. Knowledge of the true values was never necessary because the association rules would be the same regardless.

Enter filepath for DataSet:
Dataset.dat
Choose Operation:

1 to find data-to-annotation association rules.
2 to find annotation-to-annotation association rules.
3 to apply generalization rules to the Dataset.
4 to add annotations to existing rows.
5 to add annotated records to dataset.
6 to add unnanotated to dataset.
7 to see annotation suggestions.

Figure 5: Application Menu
In Figure 5 we see the main menu of the application. First we must enter the file path for the dataset file. In our case we were storing the file in the same folder as the application so only the file’s name needed to be entered. Next, we are presented with a number of options for different operations that may be performed. In this case, we are only concerned with options 1 and 2 as they are used to discover data-to-annotation and annotation-to-annotation rules respectively.

![Minimum support: .4
Minimum confidence: .8
Reading file: Dataset.dat
Finding rules with minSupport: 0.4
minConfidence: 0.8](image)

**Figure 6: Support and Confidence Entry**

After selecting either of those two options, we enter a minimum support and minimum confidence value as shown in Figure 6. Upon entering the minimum confidence value, the application begins running our apriori algorithm over the entire dataset.

![rules - Notepad](image)

**Figure 7: Association Rules Output File**
After the application finishes running and discovers all appropriate rules meeting the minimum support and confidence values, it outputs the rules as a text file as shown in Figure 7. For example, the first rule states that the presence of IDs 28 and 85 indicate the presence of Annot_1 with a confidence of 0.9659 and a support value of 0.4194.

Having implemented a way to accurately discover association rules, this method would later serve useful for verifying the accuracy of the incremental updates to association rules.
4. DISCOVERY OF CORRELATIONS

This section focuses on the discovery of annotation-related correlations. While other correlations may certainly exist, our focus lies strictly on discovering correlations which contain an annotation within the R.H.S. of the association rule. The following are the formal definitions for an annotated relation and our target correlations.

Definition 4.1 (Annotated Relation).

An annotated relation \( R \) is defined as \( R = \{ r = < x_1, x_2, \ldots, x_n, a_1, a_2, \ldots > \} \), where each tape tuple \( r \in R \) consists of \( n \) data values \( x_1, x_2, \ldots, x_n \) and a variable number of attached annotations \( a_1, a_2, \ldots, a_k \).

Definition 4.2 (Data-to-Annotation Correlations)

Given an annotated relation \( R \), a minimum support \( \alpha \), and a minimum confidence \( \beta \), the data-to-annotation correlations over \( R \) is the problem of discovering all association rules in the form

\[ x_1 x_2 \ldots x_k \Rightarrow a, \text{ where the L.H.S. is a set of data values, the R.H.S. is a single annotation, the rule's support } \geq \alpha, \text{ and the rule's confidence } \geq \beta. \]

Definition 4.3 (Annotation-to-Annotation Correlations)

Given an annotated relation \( R \), a minimum support \( \alpha \), and a minimum confidence \( \beta \), the annotation-to-annotation correlations over \( R \) is the problem of discovering all association rules in the form of: \( a_1, a_2 \ldots a_k \Rightarrow a, \text{ where the L.H.S. is a set of annotations, the R.H.S. is a single annotation, the rule's support } \geq \alpha, \text{ and the rule's confidence } \geq \beta. \)

According to these definitions, the rules we’re trying to discover must contain an annotation the R.H.S of the rule. Additionally these rules focus on the raw annotations without generalization. These
rules can be discovered with any of the state-of-art techniques such as the Apriori algorithm. The single modification made to the algorithm is the early elimination of patterns which do not contain at least one annotation value.

4.1 Generalization-Based Correlations

![Diagram of Annotation-Generalization Hierarchy]

Figure 8: Annotation-Generalization Hierarchy

Annotations can take multiple formats, which can make it difficult to discover correlations when examining just the values of raw annotations. For example, a specific type of annotation may be applied to multiple data records, but have different text for each data record. Drawing correlations based solely on the raw text value is difficult, so the process is simplified by applying generalization rules. These rules generalize the annotations to a common concept, making it possible to detect correlations that might otherwise go unnoticed. As we see in Figure 8, we can take different annotations and apply a single generalization. For example, annotations containing the words “Invalid,” “wrong,” or “incorrect” can all be generalized to the category of Invalidation.
4.1.1 Design Details

The system reads a text file containing what we refer to as “generalization rules.” This file contains the conditions for applying the generalization labels to a tuple based on the current annotations.

![Generalization Rules](image)

**Figure 9: Annotation-Generalization Rules**

The system parses a file similar to that shown in Figure 9 into a set of rules and then applies them to the dataset. The end result being that every transaction that contains Annot_1 or Annot_5 will have the Annot_X label applied to it, any transaction with Annot_4 will have Annot_Y applied, and so on. It is also important to note that a data tuple can have a given label at most once, despite the possibility of that there are multiple raw annotations mapped to the same label.

![Generalized Dataset](image)

**Figure 10: Generalized Dataset**
As seen in Figure 8, the generalized annotations are appended to the appropriate data records in the dataset. Once this extended annotated database is built, existing association rule mining techniques can be used to mine and extract the data-to-annotation and annotation-to-annotation association rules.

### 4.3 Incremental Maintenance of Correlations

In any database, there are going to be updates and additions made regularly. When dealing with increasingly large sets of data, it becomes impractical to rerun the entire association rule mining technique each time an update is made. Toward this end, we developed a system for performing incremental maintenance on the association rules. After applying an update to the database, all existing rules have their support and confidence updated, and we search for new rules that may now meet the minimum support and confidence criteria of association rules.

![Figure 11: Effect of Evolving Data on Support (S) and Confidence (C)](image)

In this project, there are three cases, indicated in Figure 11 that are considered for the incremental maintenance, each of which affect the correlations in different ways. The first case we consider is adding annotated tuples to the dataset and the second is adding un-annotated tuples. Both of these cases can already be handled by existing techniques.[1] The third case, adding annotations to existing tuples, is not currently handled by existing techniques and is the main focus of the project.
Case 1: Adding Annotated Tuples

The second case considered was the addition of annotated tuples to the existing dataset. Unlike in Case 1, there are already existing techniques in place which handle the re-evaluation and discovery of rules. Due to the addition of both data records and annotations, both data-to-annotation and annotation-to-annotation rules must be re-evaluated to ensure their validity.

Implementation

In order to add annotated tuples to the data set, we utilized the same methods as before. First we presented a prompt for the selection of the dataset to be modified. Following that the list of operations is presented and upon selecting the option for adding annotated tuples, another prompt is presented which asks for the location of the text file containing the tuples to be added.

After the selection is made, the application reads each line of the text file and writes it to the dataset already loaded by the application. After the dataset has been updated, the application rewrites the dataset file, now with the new tuples appended to the end. Once these operations are completed, the system updates the association rules. As with Case 1, the system only needs to re-evaluate rules which elements are present in the tuples. Following the updating of existing rules, the system tries to discover new rules using the same technique as in Case 1, reviewing candidate association rules which previously did not meet the minimum support and confidence requirements.

Results

To evaluate the application, we compared the results of the automated process of incrementally updating and discovering rules to the results of manually adding in annotated tuples and running the original apriori algorithm over the newly updated dataset. We found that the association rules resulting from both processes were identical, verifying the accuracy of this portion of the application.
Case 2: Adding Un-Annotated Tuples

The second case concerned the addition of unannotated data records to the dataset. For updating data-to-annotation rules, the support and confidence may only decrease, as there are only occurrences of the L.H.S. of the rule, and none of the R.H.S. Annotation-to-Annotation rules are affected much the same way except that only the support may decrease while the confidence will remain the same, unlike Cases 1 and 3, there are never going to be new rules to discover due to the lack of any annotations being added. As such, all that was necessary was to update the existing rules using the same techniques as before.

Implementation

As with all previous operations that are a part of the application, there is a prompt which asks for the location of the dataset and then presents options. Selecting option 6 prompts the entry of the filepath for the unannotated tuples that are going to be added.

The un-annotated tuples may simply be appended to the end of the dataset in question. Upon completion of this task, the system updates the rules much in the same way as in Case 1, only there is no need to search for any new rules.

Results

To evaluate the application, we compared the results of the automated process of incrementally updating rules to the results of manually adding in un-annotated tuples and running the original apriori algorithm over the newly updated dataset. We found that the association rules resulting from both processes were identical, verifying the accuracy of this portion of the application.

Case 3: Adding New Annotations

The first case considered was adding new annotations to existing data records in the database. In this case, all current data-to-annotation rules are guaranteed to remain valid because the support and
confidence of these rules cannot decrease. This also applies to annotation-to-annotation rules if the new annotation appears in the R.H.S. of the rule. In the case where the new annotation appears in the L.H.S. of the rule, however, the confidence needs to be recalculated because it is possible it will decrease and becomes lower than the minimum confidence threshold.

**Figure 12: Updating Existing Association Rules**

The algorithm depicted in Figure 12 presents the steps for updating the existing association rules after new annotations have been added. In Step 1, the data-to-annotation rules are updated. Because the de-numerator in the support and confidence of these rules doesn’t change, only the numerator values need to be re-computed. This update is performed by checking only the newly annotated data records and counting the number of new occurrences of the rule’s pattern. The number of occurrences will then be
added to the old numerator to compute the new values. Since each of these rules are guaranteed to be in the output set $U'$, they are directly copied to $U'$ after updating their support and confidence values.

In Step 2, the annotation-to-annotation rules are updated. The first For..End For loop handles annotations appearing in the R.H.S. of the rule, and operates very similarly to Step 1. The second loop handles the situation where the new annotations appear on the L.H.S of the rules. In such cases the numerator and de-numerator values of the confidence may change, either increasing or decreasing the confidence. As with Step 1, it is only necessary to count the occurrences in the newly annotated tuples to calculate the numerator and de-numerator values.
Figure 13: Discovering New Association Rules

The addition of the new annotations (the $\delta$ batch) may also create new association rules. The algorithm depicted in Figure 13 outlines the procedure of incrementally discovering the new rules. In Step 1, the new data-to-annotation rules will be discovered. First, the annotation must be a frequent annotation by itself. To perform this check efficiently, the system maintains a table containing the frequency of each annotation, and it is updated whenever a new annotation is added. If it is frequent, then from the newly
annotated tuples we extract the data value patterns that are already frequent. Notice that since the pattern is already frequent, then the de-numerator for the support and confidence of rule is already known. What is left is to compute the frequency of pattern which can be performed by checking only the data tuples in the database annotated with the added annotation. As illustrated in Figure 11, a similar procedure will be taken in Step 2, i.e., discovering the new annotation-to-annotation rules where the new annotations contribute only to the R.H.S of the rule.

Discovering the new annotation-to-annotation rules where the new annotations contribute to the L.H.S is slightly different (Step 3). This is because the de-numerator of the new rules is no longer known and it has to be computed. The procedure works by considering each new annotation, and verifying first that it is frequent. And then, from the newly annotated tuples with annotation, we extract the already-frequent annotation patterns. This pattern generates many candidate new rules. To compute the support and confidence of these rules, we need to check all data tuples in the database having annotation. This is enough to compute the support and the confidence of the rule and to verify whether or not it is a valid rule.

It is clear that the algorithm of maintaining the existing rules (Figure 10) is less expensive than that of discovering new rules (Figure 11). This is because the former requires access to only the newly annotated data tuples, whereas the latter requires access to all data tuples that have the annotation (even if the tuples are not newly annotated with it). To efficiently support the latter case, the system indexes the annotations such that given a query annotation, we can efficiently find all data tuples having this annotation. In all cases, there is no need for full database processing or re-discovering the rules from scratch.
Implementation

The annotation batch comes in the form of a text file as depicted in Figure 14. The number to the left of the colon represents the which record is to be modified, and the the annotation to the right of the colon is the new annotation being added. So in the case of the figure above, the 150th tuple, would have Annot_3 added to it.

Figure 14: Annotation Updates

1 to find data-to-annotation association rules.
2 to find annotation-to-annotation association rules.
3 to apply generalization rules to the Dataset.
4 to add annotations to existing rows.
5 to add annotated records to dataset.
6 to add unnanotated to dataset.
7 to see annotation suggestions.

Figure 15: Adding Annotations

After selecting option 4 as shown above in Figure 15, we enter the path to the file containing the updates. The application then updates the indicated data records with the proper annotations. Once the dataset is updated, the system then updates and discovers the association rules as described above.
Results

One of the main advantages of implementing the incremental maintenance of association rules is efficiency. With a large dataset, running the entire apriori algorithm each time the dataset is updated is inefficient. By storing the existing rules and candidate rules (rules slightly below the minimum support and confidence requirements) and referencing those after updates, a substantial amount of time could be saved.

Figure 16: Comparison of Run Times

As seen in the Figure 16 above, the run times to update and discover new rules is significantly faster than running the entire apriori algorithm each time an update is made. The apriori algorithm run over a dataset of approximately 8000 entries takes roughly 12 seconds each time it goes through, and that is for a conservative support and confidence value of 0.4 and 0.8 respectively. As the support value decreases the run time of the apriori algorithm takes magnitudes longer as many more potential rules need to be individually considered each time.
5. EXPLOITATION OF CORRELATIONS

As presented in Section 1, one of the goals of this work is to exploit the correlations to improve upon the quality of the annotated database. In accomplishing this goal, there are two main cases to be considered: (1) The discovery of missing annotations, and (2) The prediction of related annotations to newly inserted data records. For the first case, we created a system to scan the database and compare each tuple with the valid association rules to generate predictions. If the L.H.S pattern of a rule is present in the tuple, but the R.H.S annotation is not, then the system creates a recommendation that the R.H.S. annotation is potentially applicable to the tuple.

Figure 17: Exploitation of Correlations and Annotation-Related Recommendations.
The second case is similar to first except that the system utilizes database triggers. When a patch of new tuples is added to the database, the system automatically compares these tuples to the association rules. Much like in the first case, if the L.H.S. but not the R.H.S. of an association rule is present in a tuple, the system makes a recommendation.

In either case, the system presents only a recommendation of which annotations to add. For each prediction, the supporting association rule is displayed along with its properties, e.g., the support and confidence. Then it is up to the curators to make the final decision and add the annotation.
6. CONCLUSIONS AND FUTURE WORK

The final result of this project is a working system. Using this system, it possible to discover data-to-annotation and annotation-to-annotation rules within a dataset. In addition, a user can make additions to the dataset in the form of new annotations and new data records. The system is able to accurately and efficiently update and discover association rules as these additions are made to the dataset.

One aspect of annotation management that was not touched on in this project was the removal of annotations and data records from the dataset. The implementation of a system for handling such removals would likely be quite similar to the current updating and discovery of rules. Other future work might include implementing the incremental updating of association rules into an actual database management system, as currently it is a standalone application that must be run separately.

In conclusion, the project was a success. We developed a means of managing annotations and their association rules that proved to be both accurate and efficient.
REFERENCES