Industrial and Applications Paper

Discovering Correlations in Annotated Databases

Xuebin He Stephen Donohue Mohamed Y. Eltabakh
Worcester Polytechnic Institute, Computer Science Department, MA, USA
{xhe2, donohues, meltabakh}@cs.wpi.edu

ABSTRACT

Most emerging applications, especially in science domains, maintain databases that are rich in metadata and annotation information, e.g., auxiliary exchanged comments, related articles and images, provenance information, corrections and versioning information, and even scientists' thoughts and observations. To manage these *annotated databases*, numerous techniques have been proposed to extend the DBMSs and efficiently integrate the annotations into the data processing cycle, e.g., storage, indexing, extended query languages and semantics, and query optimization. In this paper, we address a new facet of annotation management, which is the discovery and exploitation of the hidden corrections that may exist in annotated databases. Such correlations can be either between the data and the annotations (data-to-annotation), or between the annotations themselves (annotation-to-annotation). We make the case that the discovery of these annotation-related correlations can be exploited in various ways to enhance the quality of the annotated database, e.g., discovering missing attachments, and recommending annotations to newly inserted data. We leverage the state-of-art in association rule mining in innovative ways to discover the annotation-related correlations. We propose several extensions to the state-of-art in association rule mining to address new challenges and cases specific to annotated databases, i.e., incremental addition of annotations, and hierarchy-based annotations. The proposed algorithms are evaluated using real-world applications from the biological domain, and an end-to-end system including an Excel-based GUI is developed for seamless manipulation of the annotations and their correlations.

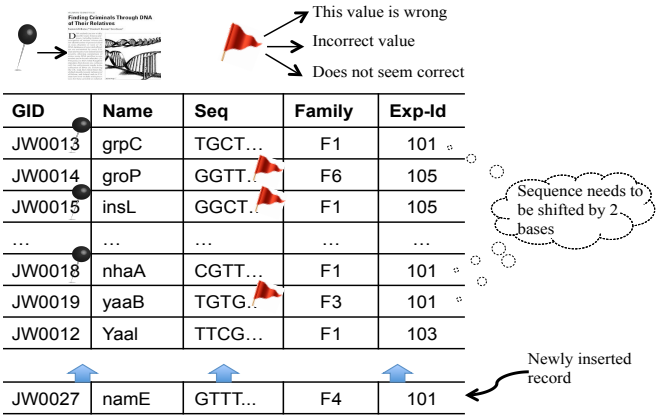
1. INTRODUCTION

Most modern applications annotate and curate their data with various types of metadata information—usually called *annotations*, e.g., provenance information, versioning timestamps, execution statistics, related comments or articles, corrections and conflict-related information, and auxiliary exchanged knowledge from different users. Interestingly, the number and size of these annotations is growing very fast, e.g., the number of annotations is around 30x, 120x, and 250x larger than the number of data records in DataBank biological database [3], Hydrologic Earth database [4, 47], and AKN ornithological database [5], respectively. Existing techniques in annotation management, e.g., [9, 15, 17, 21, 24], have made it feasible to systematically capture such metadata annotations and efficiently integrate them into the data processing cycle. This includes propagating the related annotations along with queries' answers [9, 15, 17, 24, 46], querying the data based on their attached annotations [21, 24], and supporting semantic annotations such as provenance tracking [11, 14, 20, 43], and belief annotations [23]. Such integration is vey beneficial to higher-level applications as it complements the base data with the auxiliary and semantic-rich source of annotations.

In this paper, we address a new facet of annotation management that did not receive much attention before and has not been addressed by existing techniques. This facet concerns the discovery and exploitation of the hidden correlations that may exist in annotated databases. Given the growing scale of annotated databases—both the base data and the annotation sets—important correlations may exist either between the data values and the annotations, i.e., *data-to-annotations correlations*, or among the annotations themselves, i.e., *annotations-to-annotations correlations*. By systematically discovering such correlations, applications can leverage them in various ways as motivated by the following scenarios.

Motivation Scenario 1—Discovery of Missing Attachments: *Assume the example biological database illustrated in Figure 1. Typically, many biologists may annotate subsets of the data over time—each scientist focuses only on few genes of interest at a time. For example, some of the data records in Figure 1 are annotated with a "Black Flag" annotation. This annotation may represent a scientific article or a comment that is attached to these tuples. By analyzing the data, we observe that most genes having value* `F1` *in the* `Family` *column have an attached "Black Flag" annotation. Such correlation suggests that gene* `JW0012` *is probably missing this annotation, e.g., none of the biologists was working on that gene and thus the article did not get attached to it. However, by discovering the aforementioned correlation, the system can proactively learn and recommend this missing attachment to domain experts for verification. Correlations may also exist among the annotations themselves, e.g., between the "Black Flag" and the "Red Flag" annotations. Without discovering such correlations the database may become "under annotated" due to these missing attachments.*

Motivation Scenario 2—Annotation Maintenance under Evolving Data: *Data is always evolving and new records are always added to the database. Hence, a key question is: "For the newly added data records, do any of the existing annotations apply to them?". Learning the correlations between the data and the annotations can certainly help in answering such question. For ex-*



GID	Name	Seq	Family	Exp-Id
JW0013	grpC	TGCT…	F1	101
JW0014	groP	GGTT…	F6	105
JW0015	insL	GGCT…	F1	105
…	…	…	…	…
JW0018	nhaA	CGTT…	F1	101
JW0019	yaaB	TGTG…	F3	101
JW0012	YaaI	TTCG…	F1	103
JW0027	namE	GTTT...	F4	101

Figure 1: Examples of Annotation-Related Correlations.

ample, the cloud-shaped comment in Figure 1 is attached to all data records having value `101` *in the* `Exp-Id` *column. Based on this correlation, the system can automatically predict—at insertion time—that this annotation also applies to the newly inserted* `JW0027` *tuple. Otherwise, such attachment can be easily missed and important information is lost. Clearly, delegating such task to end-users without providing system-level support—which is the state of existing annotation management engines—is not a practical assumption.*

Motivation Scenario 3—Annotation-Driven Exploration: *The discovered correlations may reveal information about the underling data that trigger further investigation or exploration by domain experts. For example, as highlighted in Figure 1, the "Red Flag" annotation semantically means invalid or incorrect data. Since these annotations can be added by different biologists and at different times, none of them may observe a pattern in the data. In contrast, by discovering (and reporting) that the "Red Flag" annotation has strong correlation with experiment id* `105`*, the domain experts may re-visit the experimental setup of this wet-lab experiment and may revise and re-validate all data generated from it.*

These scenarios demonstrate the potential gain from capturing the annotation-related correlations. Unfortunately, relying on domain experts or DB admins to manually define or capture these correlation patterns is evidently an infeasible approach. This is because the correlations may not be known in advance, hard to capture or express, dynamically changing over time, or even not 100% conformed. Moreover, the manual exploration process is error-prone, will not scale to the size of modern annotated databases, and it is a very time- and resource-consuming process. For example, the UniProt biological database has over 150 people working as full-time to maintain and annotate the database [6, 12]. Certainly, such scale of investments may not be viable to many other domains and scientific groups, e.g., it is reported in a recent science survey [49] that 80.3% of the participant research groups do not have sufficient fund for proper data curation. For these reasons, we argue in this paper that the analysis and discovery of the annotation-related associations and correlations should be an integral functionality of the annotation management engine. As a result, the correlations can be timely discovered and maintained up-to-date, and also systematic actions can be taken based on them as highlighted by the motivation scenarios.

In this paper, we investigate applying the well-known techniques of association rule mining, e.g., [8, 31, 52], to the domain of annotated databases. This is a new and promising domain for association rule mining due to the following reasons:

• Many emerging applications—especially scientific applications—maintain and rely on large-scale annotated databases [3, 5, 6]. It is reported in [1] that the *ebird* ornithological database receives more than 1.6 million annotations per month from scientists and bird watchers worldwide. These applications will benefit from the proposed techniques.

• Many annotated databases go under the very expensive and time-consuming process of manual curation, e.g., [2, 6]. The goal from this process is to ensure that correct annotations and curation information are attached to the data, and to enrich the annotations whenever possible. Nevertheless as illustrated in the motivation scenarios, the discovery of the annotation-related correlations can help in enhancing the quality of the annotated database in an automated way. And hence, reducing the effort needed in the manual curation process and freeing the domain scientists for their main task, which is scientific experimentation.

• Interestingly, annotated databases stretch the traditional techniques of association rule mining, and present new challenges as discussed in Section 3. For example, the state-of-art techniques in association rule mining fall short in efficiently handling several new cases specific to annotated databases, i.e., they cannot perform incremental maintenance of the discovered rules and they have to re-process the entire database. These cases include:

(1) *Generalization of Annotations:* Annotations can be free-text comments, which may differ in their values but have the same semantics. And hence, discovering the correlations based on the values of the raw annotations may miss important patterns. For example, referring to Figure 1, the correlation pattern involving the *Black Flag* annotation, i.e., "`Family:F1` $\Longrightarrow$ `Black Flag`" can be detected based on the raw annotation value. This is because all instances of the *Black Flag* annotation refer to the same scientific article. In contrast, for the *Red Flag* annotation, the actual annotations inserted by scientists have different values, and thus no correlation pattern can detected based on the raw values. However, by *generalizing* the annotations to a common concept—the *Red Flag* annotation in our case—we can detect the correlation patten between them and the experiment Id `105`. Therefore, building a generalization hierarchy on top of the annotations is an important step.

(2) *Integration with the Annotation Manager:* We propose to build a coherent integration between the association rule mining module and the Annotation Manager component in contrast to the *offline* mining techniques. As a result, the Annotation Manager can take informed actions based on the discovered rules, e.g., discover potential missing attachments and report them for verification (Motivation Scenario I), and annotate newly inserted data tuples with existing annotations (Motivation Scenario 2). Moreover, since the Annotation Manager cannot guarantee with 100% confidence that the predicted attachments are correct, we propose developing a verification module that enables domain experts to verify the predicted attachments.

(3) *Incremental Maintenance under Annotation Addition:* In annotated databases, the discovered correlations and association rules need to be incrementally updated under two scenarios, i.e., the addition of new data tuples, and the addition of new annotations. The former case can be handled by existing techniques that address the incremental update of association rules, e.g., [16]. These techniques assume that the new delta batch changes the size of the database, i.e., the number of data tuples increases. In contrast, in

the latter case, the new annotation batches will not change the number of data tuples, instead they change the content of the tuples—assuming the annotations are part of the tuples. Therefore, the existing incremental techniques need to be extended to handle the latter case.

In this work, we develop an end-to-end solution that addresses the above challenges in the context of a real-world application and annotation repository, which is a data warehouse for the *Caenorhabditis elegans (C. elegans)* Worm from biological sciences. To facilitate scientists' usage of the developed system, we designed an Excel-based GUI—A tool that most scientists are familiar with—through which all of the proposed functionalities can be performed.

The rest of the paper is organized as follows. In Section 2, we present the needed background, preliminaries, and our case study. In Sections 3, and 4, we present the techniques for the discovery and maintenance of the annotation-related correlations, and their exploitation, respectively. Section 5 overviews the related work while Section 6 contains the experimental evaluation. Finally, the conclusion remarks are included in Section 7.

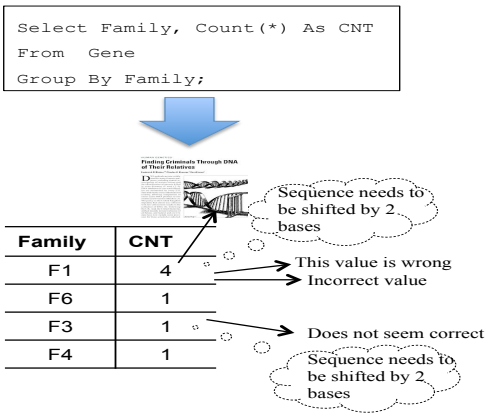


Figure 2: Automatic Propagation of Annotations at Query Time.

2. PRELIMINARIES

In this section, we give a brief overview on annotation management and association rule mining techniques, and then present our case study.

2.1 Background

Annotation Management in Relational DBs: Annotation management techniques in relational databases enable end-users to attach auxiliary information to the data stored in the relational tables [9, 15, 17, 21, 24, 46]. Annotations can be attached to individual table cells, rows, columns, or arbitrary sets and combinations of them. Some systems provide a GUI through which the annotations can be added [17, 25], while other systems extend the SQL language with new commands and clauses to enable annotation addition [17, 21, 25]. For example, the work in [21] introduces a new `Add Annotation` command to SQL as follows:

```
Add Annotation
[Value <text-value> | Link  <path-to-file>]
On <sql-statement>;
```

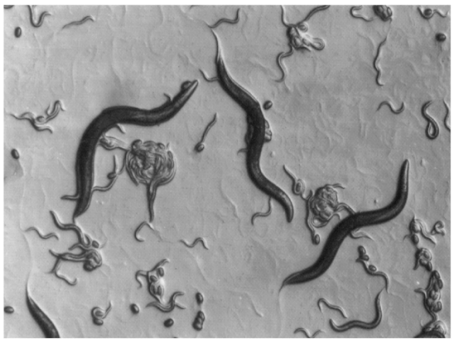
This command will first trigger the execution of the specified *<sql-statement>* to identify the data tuples and the attributes to which the annotation will be attached. The annotation can be provided as a text value (using the `Value` clause), or as a link to a file (using the `Link` clause). In all of these systems, the organization of annotations, i.e., storage scheme and indexing, is fully transparent to end-users.

At query time, when a standard SQL query is submitted, the underlying database engine will not only compute the data tuples involved in the answerset, but will also compute the related annotations that should be reported along with the answerset. This is not straightforward since the data may go though complex transformations, e.g., projection, join, grouping and aggregations. Therefore, the semantics of the query operators have been extended to manipulate the data as well as their attached annotations in a systematic way. For example, one possible semantic is to union the annotations when performing grouping, joining, or duplicate elimination over a group of tuples. According to this semantic, the SQL query in Figure 2 produces the illustrated output—assuming duplicate annotations on the same tuple are eliminated.

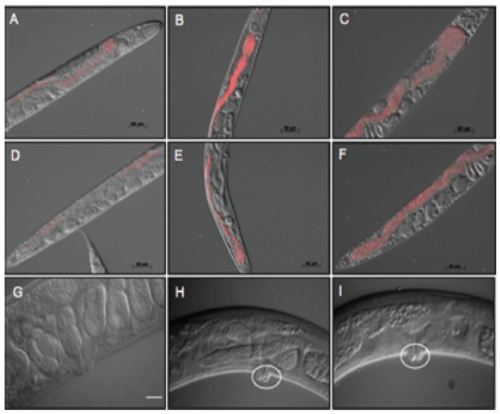
Since in large-scale annotated databases, the number of reported annotations on a single output tuple can be large, the work in [32, 50] proposed summarizing the annotations using data mining techniques, e.g., clustering, classification, and text summarization, and reporting the summarize instead of the raw annotations. The proposed work of discovering the correlations in annotated databases is complementary to the existing techniques and can be integrated with any of the existing systems.

Association Rule Mining: Association rule mining is a well-known problem in data mining that concerns the discovery of correlation patterns within large datasets [8, 44, 45, 52]. An association rule in the form of "$\texttt{X} \Longrightarrow \texttt{Y}$, `support` $= \alpha$, `confidence` $= \beta$" means that the presence of the L.H.S itemset `X` implies the presence of the R.H.S itemset `Y` (in the same transaction or tuple) with support equals to α and confidence equals to β. Typically, $X \cap Y = \phi$, and the support is computed as the fraction of transactions (or tuples) containing $X \cup Y$ relative to the database size, while the confidence is computed as $support(X \cup Y)/support(X)$. Therefore, given a minim support *min_supp* and minimum confidence *min_conf*, the association rule mining technique discovers all rules having support and confidence above the specified *min_supp* and *min_conf*, respectively.

It has been observed in many real-world applications, that the number of generated rules can be very large and many of them may not be interesting. Therefore, additional measures have been proposed in [31, 44], which include the *lift* and *conviction* measures. The former is computed as $\frac{support(X \cup Y)}{support(X) * support(Y)}$, and the latter is computed as $\frac{1-support(Y)}{1-confidence(X \Longrightarrow Y)}$. The higher these measures, the more interesting the rule. Another related extension to the standard association rule mining problem is the mining of multi-level rules [44, 45]. 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. Association rule mining has numerous applications in various domains including market analysis, biology, healthcare, environmental sciences, and beyond [31]. The proposed work extends these applications to the emerging domain of annotated databases.



(a) Caenorhabditis elegans. Lives in garden soil and feeds on bacteria, e.g., E. coli.



(b) Time course of intestinal distention in *C. elegans* worms exposed to *S. cerevisiae*. Worms were exposed to RFP-marked, wildtype yeast from hatching and photographed on day 3 (A and D), day 4 (B and E), and day 5 (C and F). The experiment was done three times, and 60 to 75 worms were observed over a 3-day period. (A to C) Anterior region of the worm; (D to F) posterior region of the worm. Accumulation of yeast began in the pharynx region (compare panels A and D) and proceeded to the posterior. (G to I) *S. cerevisiae* also induces vulval swelling in the worms. Worms exposed to *S. cerevisiae* (H to I) show abnormal vulval swelling (white circles) compared to the control sample grown on *E. coli* (G). Bars: A to G, 50 µm; H and I, 20 µm.

Figure 3: Case Study: Building Data-Annotation Repository for Caenorhabditis elegans (C. elegans) Worm.

2.2 Case Study: Annotated Repository for C. elegans Worm

Although the proposed work is applicable to annotated databases in general, we consider one case study as an example. This case study focuses on building a database repository for the *Caenorhabditis elegans (C. elegans) worm*, which integrates data from multiple databases and labs studying the genetics and fungal pathogenesis of the organism (Refer to Figure 3). Some of these sources maintain relational databases, while other use excel sheets to store their data. Each of the sources maintains various types of curation information and annotations related to the data records, e.g., images, publications, observations, corrections, and experimental setups. In these data sources the curation information is not modeled as *annotations*. Instead, they are modeled as regular data with relationships to the data tuples, e.g., in the relational databases, the images and observations are stored in separate tables linked to the primary data tables through PK-FK constraints.

The disadvantage of this modeling scheme, i.e., modeling the annotations as data, is that applications lose the benefits of annotation management. That is, annotation management tasks are entirely delegated to end-users and higher-level applications starting from the storage and indexing of annotations and ending by explicitly encoding the propagation semantics within each of the users' queries. Both tasks have been shown to be very complex and sophisticated. For example, the storage and indexing mechanisms need to deal with the combinatorial relationship between annotations and data, e.g., annotations can be attached to single table cells (attributes), rows, columns, arbitrary sets and combinations of them, or even attached to sub-attributes [21, 24]. Moreover, manually encoding the annotations' propagation within each query is not only error-prune, and lacks optimizations, but also renders even simple queries very complex [9, 13, 26, 46]. That is why annotation management engines have been proposed to efficiently and transparently manage such complexities across applications.

To address the above limitations, we opt for leveraging our previous systems and work in annotation management [21, 50] to model the metadata information as annotations. These systems are built on top of PostgreSQL DBMS, and thus the repository will acquire the benefits of both a DBMS and an annotation management engine. We developed an Excel-based user interface to enable seamless visualization of the data as well as the annotations (See Figure 4). For the purpose of this project, the annotations are categorized into three basic types, which are: (1) Regular comments or observations, which covers any free-text values, (2) Articles and

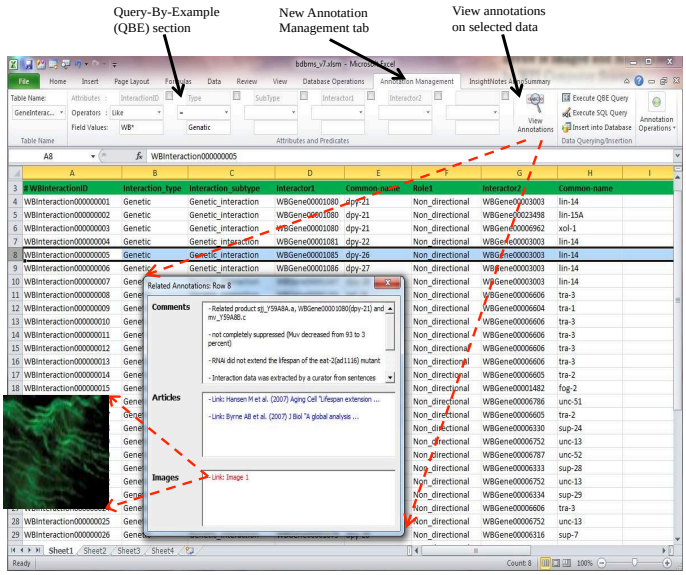


Figure 4: Excel-Based GUI for Annotation Management.

documents, and (3) Images. In the Excel-based GUI, scientists can query their data either by writing direct SQL queries, or thought a QBE interface as illustrated in Figure 4. Then, they can select specific rows or table cells of interest, and click the *View Annotation* button, which reports the annotations related to the selected datasets in a new window. This window has three sections for the three annotation types mentioned above as shown in the figure. For the articles and images, they are uploaded to a web-server and can be opened by clicking the corresponding link.

3. DISCOVERY OF CORRELATIONS

Given an annotated database, the primary challenge is to discover and incrementally maintain the hidden annotation-related correlations within the database, which is the focus of this section. The basic unit in an annotated database is an *"annotated relation"*. In the following, we formally define an annotated relation and the target correlations.

Definition 3.1 (Annotated Relation)**.** *An annotated relation* $\mathcal{R}$ *is defined as* $\mathcal{R} = \{r =< x_1, x_2, ..., x_n, \mathbb{a}_1, \mathbb{a}_2, \mathbb{a}_3, ... >\}$*, where*

each data tuple $r \in \mathcal{R}$ consists of n data values $x_1, x_2, ..., x_n$, and a variable number of attached annotations $\mathtt{a}_1, \mathtt{a}_2, ..., \mathtt{a}_k$.

Definition 3.2 (Data-to-Annotation Correlations)**.** *Given an annotated relation $\mathcal{R}$, a minimum support α, and a minimum confidence β, the data-to-annotation correlations over $\mathcal{R}$ is the problem of discovering all association rules in the form of: $x_1{:}v_1$, $x_2{:}v_2$, ..., $x_k{:}v_k \Longrightarrow \mathtt{a}$, where the L.H.S is a set of column names (x_i) and corresponding data value (v_i), the R.H.S is a single annotation, the rule's support $\geq \alpha$, and the rule's confidence $\geq \beta$.*

Definition 3.3 (Annotation-to-Annotation Correlations)**.** *Given an annotated relation $\mathcal{R}$, a minimum support α, and a minimum confidence β, the annotation-to-annotation correlations over $\mathcal{R}$ is the problem of discovering all association rules in the form of: $\mathtt{a}_1\ \mathtt{a}_2\ ...\ \mathtt{a}_k \Longrightarrow \mathtt{a}$, where the L.H.S is a set of annotations, the R.H.S is a single annotation, the rule's support $\geq \alpha$, and the rule's confidence $\geq \beta$.*

According to Definitions 3.2 and 3.3, the rules to be discovered must involve an annotation in the R.H.S of the rule. In addition, these rules focus on the raw annotations without generalization. This is applicable especially for annotations of type *image* or *publication*, where a single image or publication can be attached to many data tuples. Discovering these rules is straightforward using any of the state-of-art techniques, e.g., the A-priori [8], or FP-Tree [30] algorithms. The only modification that we introduced to these algorithms is the early elimination of candidate patterns that do not include at least one annotation value.

3.1 Incremental Maintenance of Correlations

An annotated database may evolve and change in three different ways, which are: (1) Adding new un-annotated data tuples (refer to it as $\Delta_{unannotated}$), (2) Adding new annotated data tuples (refer to it as $\Delta_{annotated}$), and (3) Adding new annotations to existing data tuples (refer to it as δ). Each of these changes may affect the discovered association rules as summarized in Figure 5. The maintenance of association rules under incremental data updates has been studied in existing work [16, 41]. However, these existing techniques can handle only the first two cases mentioned above, i.e., the $\Delta_{unannotated}$ and $\Delta_{annotated}$ cases, but not the third case, i.e, the δ case. The reason is that the assumption in these techniques is that the number of data tuples change (get increased), which is true for the first two cases. In contrast, in the third case, the number of the data tuples is fixed, but their content changes due to the addition of new annotations.

Figure 5 summarizes the three cases mentioned above and their effect on the association rules. The $\Delta_{unannotated}$ case does not add any new rules since no new annotations are added. For the existing rules, both the support and confidence of the data-to-annotation rules may get decreased and need to be re-computed (Column 2 in Figure 5). Nevertheless, for the annotation-to-annotation rules, only the support may decrease and need to be re-computed, but the confidence remains unchanged (Column 3 in Figure 5). The $\Delta_{annotated}$ case may introduce new association rules since the new data tuples are annotated. Moreover, all of the existing rules may get affected positively or negatively. For these two cases, the existing techniques in [29, 44] can be directly applied to efficiently and incrementally update the association rules.

The third case—which is not handled by existing techniques—concerns the addition of new annotations to existing data tuples. For this case, all existing data-to-annotation rules are guaranteed to remain valid because the support and confidence of these rules

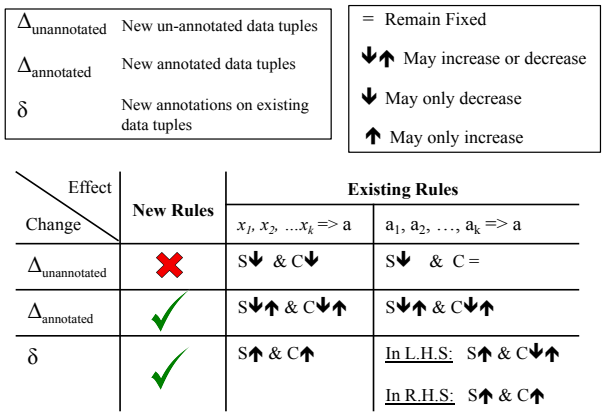


Figure 5: Effect of Evolving Data on Support (S) and Confidence (C).

cannot be decreased. The same intuition applies to the annotation-to-annotation rules if the new annotation appears in the R.H.S of the rule, i.e., the support and confidence may only increase. However, if the new annotation appears in the L.H.S of a rule, then the confidence of this rule needs to be re-computed because it may decrease and becomes below the *min_conf* threshold. Finally, this third case may introduce new association rules that need to be discovered as indicated in Figure 5. In the following, we present a pseudocode on how these changes take place incrementally.

The algorithm depicted in Figure 6 presents the main steps of updating the existing rules. In Step 1, the data-to-annotation rules are updated. Basically, the denominator in the support and confidence of these rules does not change, and thus only the numerator values need to be re-computed. This update can be performed by checking only the newly annotated data tuples and counting the number of new occurrences of the rule's pattern (L.H.S $\cup$ R.H.S). This count will be added to the old numerator to compute the new values. Since all 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 the case where the new annotations do not appear in the L.H.S of an existing rule, but appear on the R.H.S. This case is very similar to Step 1, where all the rules will get their support and confidence updated (only the numerator values), and then copied to the output set U'. The second `For...End For` loop handles the case where the new annotations appear on the L.H.S of the rules. in this case both the numerator and the denominator values of the confidence may change and hence, it may increase of decrease. Fortunately, updating these values can be also performed by only checking only the newly annotated data tuples and counting the number of new occurrences that will be added to either of the numerator or denominator values. Depending on the new confidence, if the rule is still valid, then it will be copied to the output set U'. It is worth highlighting that in updating the existing association rules (Steps 1 & 2 in Figure 6), we only need to process the newly annotated data tuples without touching the rest of the database.

The addition of the new annotations (the δ batch) may also create new association rules. The algorithm depicted in Figure 7 outlines the procedure of incrementally discovering the new rules. In Step 1, the new data-to-annotation rules in the form of $x_1\ x_2\ ...\ x_k \Longrightarrow \mathtt{a}$ will be discovered, where $\mathtt{a} \in \delta$. First, $\mathtt{a}$ must be a frequent an-

Case III (δ Addition): Updating Existing Association Rules

Input:
- New annotations delta δ = {a_1, a_2, …, a_n}
- Set of existing association rules U = {u_1, u_2, …, u_m}
- Original DB

Output:
- Updated set of association rules U'
- New DB' = DB + δ

Step1: Existing Data-to-Annotations rules
- **For** (each rule u_i having annotation "*a*" && δ in R.H.S) **Loop**
  - // Update the support and confidence of u_i
  - T ← The data tuples newly annotated with "*a*"
  - Based on T update the u_i.supp & u_i.conf
  - Copy u_i to U' // u_i is guaranteed to be valid
- **End For**

Step2: Existing Annotation-to-Annotations rules
- **For** (each rules u_i where none of δ appears in L.H.S & "*a*" && δ in R.H.S) **Loop**
  - // Update the support and confidence of u_i
  - T ← The data tuples newly annotated with "*a*"
  - Based on T update the u_i.supp & u_i.conf
  - Copy u_i to U' // u_i is guaranteed to be valid
- **End Loop**

- δ' ← annotations in δ that appear in L.H.S of some rules
- **For** (each rule u_i where any of δ' appears in L.H.S) **Loop**
  - // Update the support and confidence of u_i
  - T ← The newly annotated data tuples having an annotation from δ'
  - Based on T update the u_i.supp & u_i.conf
  - **If** (u_i is still a valid rule) **Then**
    - Copy u_i to U'
  - **End If**
- **End Loop**

Figure 6: Case III (δ Addition): Updating Existing Rules.

notation 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 $\mathtt{a}$ is frequent, then from the newly annotated tuples, denoted as T, we extract the data-value patterns that are already frequent, say $x_1\ x_2\ ...\ x_k$. Notice that since $x_1\ x_2\ ...\ x_k$ is already frequent, then the denominator for the support and confidence of rule $x_1\ x_2\ ...\ x_k \Longrightarrow \mathtt{a}$ is already known. What is left is to compute the frequency of pattern $x_1\ x_2\ ...\ x_k, \mathtt{a}$, which can be performed by checking only the data tuples in the database annotated with $\mathtt{a}$. As illustrated in Figure 7, a similar procedure will be taken in Step 2, i.e., discovering the new annotation-to-annotation rules where the new annotations $\mathtt{a} \in \delta$ contribute only to the R.H.S of the rule.

Discovering the new annotation-to-annotation rules where the new annotations $\mathtt{a} \in \delta$ contribute to the L.H.S is slightly different (Step 3). This is because the denominator of the new rules is no longer known and it has to be computed. The procedure works by considering each new annotation $\mathtt{a} \in \delta$, and verifying first that it is frequent (if not, then the process stops). And then, for each data tuple t that is receiving $\mathtt{a}$ as a new annotation, we extract the already-frequent annotation patterns, say $p = \mathtt{a}'_1, \mathtt{a}'_2, ..., \mathtt{a}'_k$, i.e., p is already frequent and attached to t. By augmenting $\mathtt{a}$ to p, we generate several candidate new rules in the form of $\mathtt{a}'_1, \mathtt{a}, ..., \mathtt{a}'_k \Longrightarrow \mathtt{a}'_i$. Notice that $\mathtt{a}$ can be a new annotation over tuple t, but it is an already-existing annotation over many other tuples in the database. Therefore, to compute the support and confidence of these rules, we need to check all data tuples in the database having annotation $\mathtt{a}$. 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 6) is less expensive than that of discovering new rules (Figure 7). 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 annotation $\mathtt{a} \in \delta$ (even if the tuples are not newly annotated with $\mathtt{a}$. 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.

Case III (δ Addition): Discovering New Association Rules

Input:
- New annotations delta δ = {a_1, a_2, …, a_n}
- Original DB

Output:
- New set of association rules U'
- New DB' = DB + δ

Step1: New Data-to-Annotations rules $x_1, x_2, …, x_k \Rightarrow a_i$
- **If** (a_i is frequent) **Then**
  - T ← the data tuples newly annotated with a_i.
  - Find patterns "$x_1, x_2, …, x_k$" within T that are already frequent.
  - Check if $x_1, x_2, …, x_k \Rightarrow a_i$ is valid
    - De-numerators of the support and confidence are known
    - Compute the frequency of pattern "$x_1, x_2, …, x_k, a_i$" by checking only the data tuples annotated with a_i
    - Now the support and confidence of the rule can be verified.
    - If the rule is valid ➔ add it to U'
- **End If**

Step2: New Annotation-to-Annotations rules $a'_1, a'_2, …, a'_k \Rightarrow a_i$
- **If** (a_i is frequent) **Then**
  - T ← the data tuples newly annotated with a_i
  - Find patterns "$a'_1, a'_2, …, a'_k$" within T that are already frequent.
  - Check if $a'_1, a'_2, …, a'_k \Rightarrow a_i$ is valid
    - De-numerators of the support and confidence are known
    - Compute the frequency of pattern "$a'_1, a'_2, …, a'_k, a_i$" by checking only the data tuples annotated with a_i
    - Now the support and confidence of the rule can be verified.
    - If the rule is valid ➔ add it to U'
- **End If**

Step3: New Annotation-to-Annotations rules where a_i may appear in L.H.S of the rule
- **For** (each frequent annotation patterns "$a_1, a_2, …$" from δ) **Loop**
  - T ← the data tuples newly annotated with "$a_1, a_2, …$"
  - Find patterns "$a'_1, a'_2, …, a'_k$" within T that are already frequent.
  - Create candidate rules where subset (or all) of "$a_1, a_2, …$" appear in the L.H.S of the rule
  - Compute the support and confidence by checking only the data tuples annotated with "$a_1, a_2, …$"
  - If the rule is valid ➔ add it to U'
- **End Loop**

Figure 7: Case III (δ Addition): Discovering New Rules.

3.2 Generalization-Based Correlations

Generalizing the raw annotation values to higher concepts may lead to discovering important association rules that cannot be discovered from the raw values. The *Red-Flag* annotation in Figure 1 is a good example of this case. The reason this case is important in annotated databases is that the annotations can be added by many curators, and they may not follows specific ontology. And thus, multiple annotations may carry the same semantics but differ in their raw values. There are extensions to association rule mining techniques that can discover the rules under the presence of a generalization hierarchy [29, 44, 45]. However, these techniques assume that the hierarchy, and the assignments between the raw values to

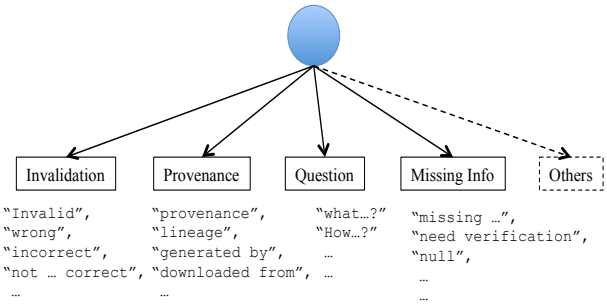


Figure 8: Example of Annotation-Generalization Hierarchy.

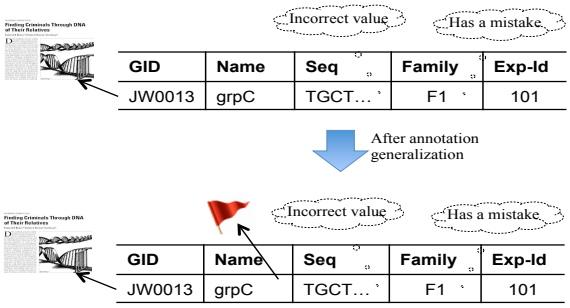


Figure 9: Applying Annotation-Generalization over Example Tuple.

the hierarchy elements are given as input, which is usually not the case in annotated databases.

In this work, we assume the more practical case where the domain experts know the generalization hierarchy, i.e., the generic types of annotations of interest, but the raw annotations are not yet labeled. An example of this hierarchy is illustrated in Figure 8, where several types can be of interest, e.g., *"Invalidation"* (comments that highlight errors or invalid values), *"Provenance"* (comments capturing the source of data or how it is generated), *"Question"* (comments including questions), and *"Missing Info"* (comments highlighting missing values or more investigation). The hierarchy will always include a separate class, called *"Others"* to which any un-generalized annotation will belong.

In general, any text classification technique can be used. As a proof of concept, we use the technique proposed in [18]. Since we assume no training set or classifier model is given, the model is first created as follows:

```
1- Select α real annotations at random and manually
   label them to build the 1st classifier model.

2- Select β% of the real annotations at random and
   classify them using the trained classifier.

3- Manually verify the results, and re-label the
   wrong classification to refine the model.

4- Repeat Steps 2 & 3 until achieving an acceptable
   accuracy.
```

In Step 1, in addition to manually labeling an initial set of α annotations, we also create synthetic annotations to each class capturing the keywords in that class. For example, as highlighted in Figure 8, keywords like *"wrong"*, *"incorrect"*, *"invalid"* are embedded in synthetic annotations under the 1^{st} class label, while keywords like *"source"*, *"generated from"* are added under the 2^{nd} class label. The creation process iterates between labeling and verifying a small subset of annotations ($\beta\%$) until the classier reaches an acceptable accuracy (Step 4).

After building the classifier model, it is applied over all annotations in the dataset, and data tuples get annotated with the class labels corresponding to their raw annotations—Except for "Others" label for which its annotations are not generalized. A data tuple can have a given label at most once even if there are multiple raw annotations mapping to the same label. This is the same model used in association rule mining techniques that handle a generalization hierarchy [29, 44, 45]. For example, the top data tuple in Figure 9 has two attached annotations classified into the *"Invalidation"* label, i.e., the *Red Flag*, and thus after attaching the classification decision, it generates the tuple at the bottom of the figure.

After building the extended annotated database, existing techniques can mine and extract the data-to-annotation association rules in the form of: $x_1\ x_2\ ...\ x_k \Longrightarrow \mathbb{b}$, where $\mathbb{b}$ is either a raw annotation or an annotation's class label. The same applies for annotation-to-annotation rules, which are in the form of: $\mathbb{b}_1\ \mathbb{b}_2\ ...\ \mathbb{b}_k \Longrightarrow \mathbb{b}$, where none of the L.H.S or R.H.S values (either raw or class label values) have an ancestor-descendant relationship in the generalization hierarchy.

4. EXPLOITATION OF CORRELATIONS

As discussed in Section 1, one of our goals is to exploit the discovered correlations to enhance the quality of the annotated database. We consider two exploitation scenarios: (1) The prediction of related annotations to newly inserted tuples (Motivation Scenario 2), and (2) The discovery of missing attachments when new association rules are found (Motivation Scenario 1).

Insertion of New Data Tuples: For this case, an automatic database trigger (at *After Insert at Row Level*) is created for each database table. The trigger forwards a newly inserted tuple t to the Annotation Manager, which checks t against the available association rules. If the L.H.S pattern of a rule is present in t without the R.H.S annotation, then the system creates a recommendation that the R.H.S annotation is potentially applicable to t. For example, referring to the motivation example in Figure 1, the following rule can be derived from the data:

`Exp-Id:101` $\Longrightarrow$ `"Sequence need to be shifted by 2 bases."`

The newly added tuple `JW0027` contains the L.H.S of the rule, and hence the system predicts and creates a recommendation that the R.H.S annotation may be related to the new tuple. The end-user will be notified that the system has created annotation predictions for the new tuple, and the prediction(s) will be stored in a system table along with its supporting rule (the rule generated the recommendation) for later verification and approval as explained in sequel.

To enable efficient searching for the association rules matching the newly inserted tuple, the L.H.S of the rules are indexed. Since the L.H.S may contain several pairs in the form of `columnName:value`, we first itemize and store the L.H.S in a normalized form, i.e., one record for each pair, and then index them using a B-Tree index. In this case, given a new data tuple, the *After Insert* database trigger will create lookup keys from the new tuple, i.e., each column name and its value will form one lookup key, and then search the normalized table to find a "superset" of the candidate rules. The tuple will be checked against this candidate set to find the actual matches.

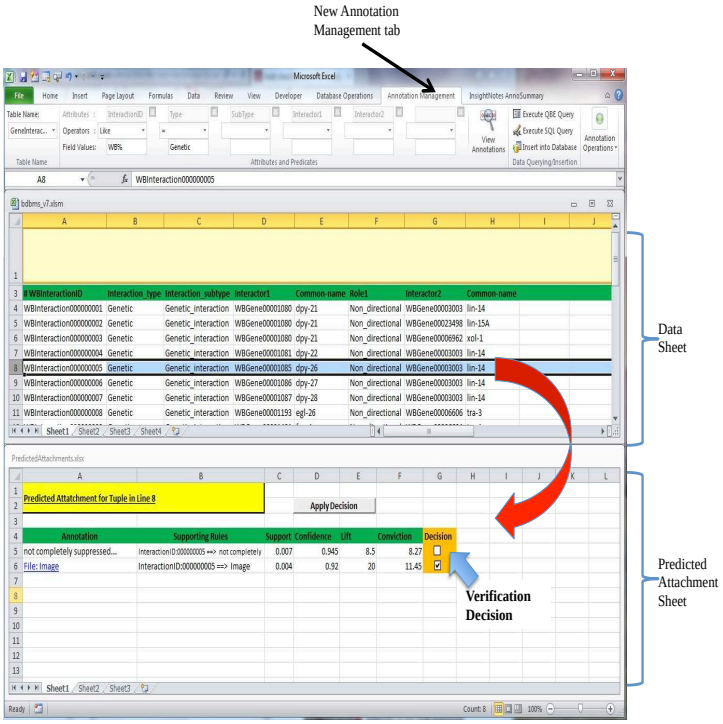


Figure 10: Exploitation of Correlations and Annotation-Related Recommendations.

Updating the Association Rules: As presented in Section 3.1, the annotation-related association rules may change periodically when new batches of data or annotations are applied. The change may take three forms: (1) Valid rules remain valid, (2) Valid rules become invalid, and (3) New rules are discovered. In the 1^{st} case, nothing will change in the system. In the 2^{nd} case, the pending recommendations awaiting verification whose supporting rules become invalid will be eliminated. This is because the system will not recommend attachments without rules supporting available as evidences. [1] In the 3^{rd} case, the discovery of new rules will trigger the system to search for data tuples matching the L.H.S of the new association rules, but missing the R.H.S annotation in the rule. If a matching is found, then a new pending recommendation will be added to the system table. Given a newly discovered rule having a L.H.S in the form of: "$C_1 : v_1,\ C_2 : v_2,\ \ldots,\ C_m : v_m$", where C_i is a column name and v_i is the column's value, the searching for the matching data tuples is performed as follows. A query on the same table to which the new rule is related is formed, and it consists of a set of conjunctive predicates in the form of $C_i = v_i\ \forall\ 1 \leq i \leq m$. The returned tuples will be then checked if they are missing the R.H.S of the rule, i.e., the annotation value, and if so, then a new recommendation is generated.

It is worth highlighting that the number of discovered rules by considering solely the *support* and *confidence* thresholds is in most cases very high [31, 44]. This will result in many uninteresting predictions and recommendation. To overcome this problem, we integrate additional properties such as *lift* and *conviction* measures that measure how interesting the rule is (See Section 2). That is, among all the rules that satisfy the support and confidence requirements, we use in the exploitation process only a subset of these rules that also satisfy the lift and conviction requirements.

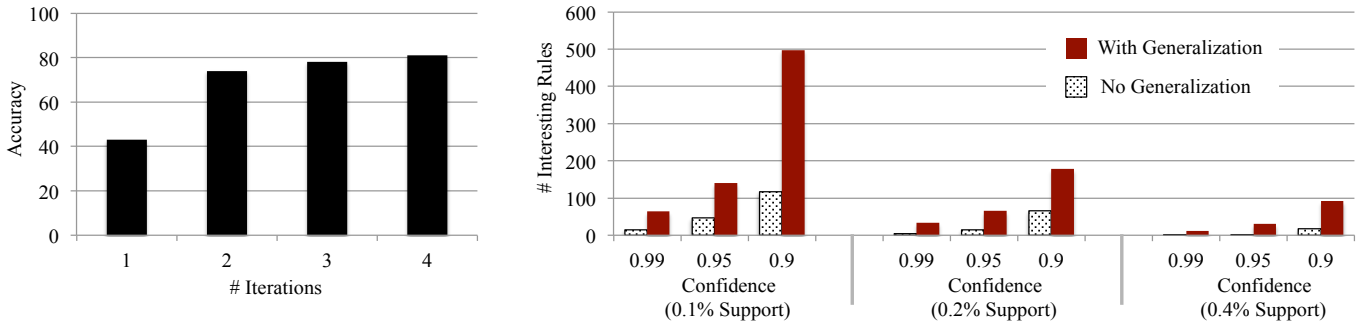
Manipulation Interface: To seamlessly enable the verification process, we extend the Excel-based GUI presented in Section 2.2 such that domain experts can visualize the recommendations from the tool and decide whether or not each prediction will be accepted (See Figure 10). The tool enables reporting and visualizing the pending predictions either by providing a database table name, or by specifying a select statement to limit the scope of interest. The reported predictions can be then sorted according to various criteria, e.g., the confidence of the association rule suggested the prediction. As Figure 10 illustrates, the data tuples from a given table (or a select statement) are reported on the top-level excel sheet, and then when a tuple is selected, its related predictions and recommendations are dynamically reported on the bottom-level sheet. For each prediction, the supporting association rule is displayed along with its properties, e.g., the support, confidence, lift, and conviction. Curators can then use the checkbox highlighted in the figure to decide on whether or not to accept the recommendation.

5. RELATED WORK

Annotation management is widely applicable to a broad range of applications, yet it gained a significant importance within the context of scientific applications [27, 35, 40]. Therefore, to help scientists in their scientific exterminations and to boost the discovery process, several generic annotation management frameworks have been proposed for annotating and curating scientific data in relational DBMSs [9, 17, 24, 25, 26, 46]. Several of these systems, e.g., [9, 17, 24, 46], focus on extending the relational algebra and query semantics for propagating the annotations along with the queries' answers at query time. The techniques presented in [46] address the annotation propagation for containment queries, while the techniques [13] address the propagation in the presence of logical database views. They address, for example, the minimum amount of data that need to annotated in the base table(s) in order for the annotation to appear (propagate) to the logical view. The work in [21] proposed compact storage mechanisms for storing multi-granular annotations at the raw-, cell-, column-, and table-levels, as well as defining behaviors for annotations under the different database operations. Moreover, the techniques proposed in [21, 36] enable registering annotations in the database system and automatically applying them to newly inserted data tuples if they satisfy pre-defined predicates.

Other systems have addressed special types of annotations, e.g., [15, 23]. For example, the work in [15] have addressed the ability to annotate the annotations, and hence they proposed a hierarchal approach that treats annotations as data. On the other hand, the BeliefDB system in [23] introduced a special type of annotations, i.e., *the belief annotation*, that captures the different users' beliefs either about the data or others' beliefs. In both systems, the query engine is extended to efficiently propagate/query these annotations. It have been also recognized in [28, 34] that annotations may have semantics and based on these semantics the propagation in the query pipeline may differ, e.g., instead of getting the union of annotations under the join operation, getting the intersection makes more sense under some annotation semantics. In our previous work [50], we addressed the challenge of managing number of annotations that can be orders of magnitude larger than the number of base data tuples. In this case, reporting the raw annotations will be overwhelming and useless. Instead, we proposed summarizing the annotations into concise forms, and then proposed an extended query engine to efficiently propagate these summaries.

[1] Recommended attachments that have been approved by the end-users will remain in the system even if their supporting rules become invalid at a later point in time. This is because an approved attachment is viewed as a correct and permanent one.



(a) Accuracy of the Generalization Algorithm.

(b) Effect of Generalization on the Generated Rules

Figure 11: Annotation Generalization and Rule Generation.

Although the above systems provide efficient query processing for annotations, none of them have addressed the facet of mining the rich repositories of annotations to discover interesting patterns, e.g., discovering the annotation-related correlations proposed in this paper. Therefore, the proposed work is complementary to existing systems and creates an automated mechanism for enhancing the quality of annotated databases, which is currently handled through a manual curation process [2, 6]. In this process, domain experts manually curate the data, ensure correct and high-quality annotations are attached to the data, and potentially add or remove further attachments between the annotations and the data. Certainly, this curation process is very time consuming, error-prune and does not scale well, and more importantly consumes valuable cycles from domain experts and scientists. With the proposed work, a significant effort in discovering missing attachments and relationships between the data and the annotations can be automated.

Annotations have been supported in contexts other than relational databases, e.g., annotating web documents [10, 33, 37, 51], and pdf files [38, 39, 48]. These techniques focus mostly on annotating different pieces within a document (or across documents) with useful information, e.g., extracting the key objects or providing links to other related objects. In the domains of e-commerce, social networks, and entertainment systems [22, 42], the annotations are usually referred to as *tags*. These systems deploy advanced mining and summarization techniques for extracting the best insight possible from the annotations to enhance users' experience. They use such extracted knowledge to take actions, e.g., providing recommendations and targeted advertisements [7, 19, 42]. However, none of these systems focus on mining and discovering correlations within the annotation repositories.

6. EXPERIMENTS

Setup and Dataset: The experiments are performed using our annotation management engines [21, 50], which are based on the open-source PostgreSQL DBMS. The experiments are conducted using an AMD Opteron Quadputer compute server with two 16-core AMD CPUs, 128GB memory, and 2 TBs SATA hard drive. Our objectives are: (1) To quantify the effectiveness of the annotation generalization technique, (2) The performance of discovering the annotation-based association rules over static data, and (3) The efficiency of the proposed incremental techniques to update the rules under new batches of annotations. The experimental dataset represent the C. elegans repository, which we are building on site. The repository consists of 27 database tables, where the main table of our focus is the `Gene` table that contains approximately 17,500 genes integrated from multiple sources. The table has nine columns, e.g., *Gene Id, Locus, Strain, Stage, Location*, and *Length*. The table has a total number of 43,000 publications attached to the genes in addition to other 8,120 free-text comments. These publications and comments represent the annotations attached to the `Gene` table. In the dataset, each record has between 0 annotations (the minimum) and 16 annotations (the maximum).

In Figure 11, we study the effect of annotation generalization on the generation of interesting annotation-related association rules. Figure 11(a) illustrates the accuracy of the generalization algorithm presented in Section 3.2. The x-axis indicates the number of iterations from 1 to 4, while the y-axis represents the obtained accuracy from the manual verification step (Step 3 of the algorithm). As the figure shows, there is a big jump in accuracy from Iteration 1 to Iteration 2, and then as more iterations are performed the accuracy slightly increases. This is mostly because our generalization model is simple and consists of one level only. However, as the generalization model becomes more complex, we expect to more iterations will be needed to reach the desired accuracy. In the subsequent experiments, we will use the results obtained after performing 4 iterations.

In Figure 11(b), we present the number of interesting association rules discovered in the entire dataset under varying confidence and support degrees (the x-axis). Since the expected rules are not necessarily globally frequent, we use very low support, e.g., 0.1% to 0.4%, while setting the confidence to a very high threshold as indicated in the figure. As expected, as the support or confidence thresholds increase, less number of interesting rules are discovered. The number of discovered rules is relatively manageable and not very large because we also enforce minimum lift and conviction thresholds to narrow down the reported rules to the strongest ones only. Both thresholds are set to value 5. In the remaining experiments, we will consider the dataset under the annotation generalization case, i.e., the annotations have been generalized to enable the discovery of more rules.

In Figure 12, we study the execution time of discovering the annotation-related rules in the entire dataset. We vary the confidence and support thresholds as depicted in the figure. In this experiment that dataset is static, i.e., there are no new batches of data or annotations. As the figure illustrates the execution time ranges from 8 secs to 22 secs depending on the used thresholds. We have also studied the execution time of the mining algorithm without annotation generalization and the observed differences are not significant, e.g., the technique without generalization are faster by 1% to 7% compared to the other case.

In Figure 13, we study the execution time under the addition of new annotation batches. We focus on the 3^{rd} case presented in

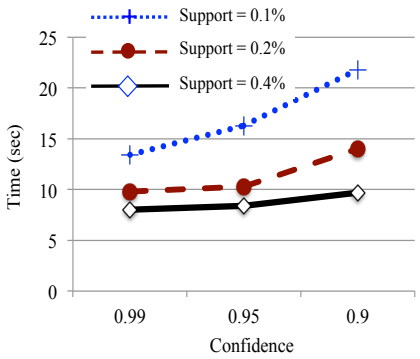


Figure 12: Evaluation of Execution Time.

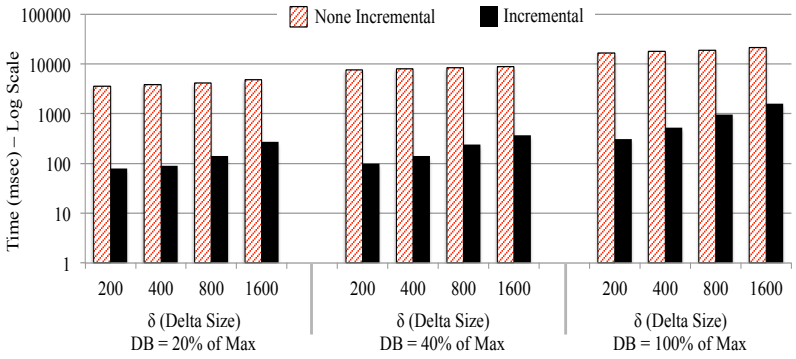


Figure 13: Execution Time of Incremental vs. Non-Incremental Algorithms (Case III).

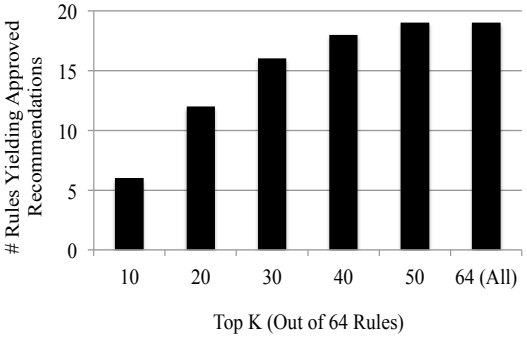


Figure 14: Number of Rules Yielding Approved Recommendations.

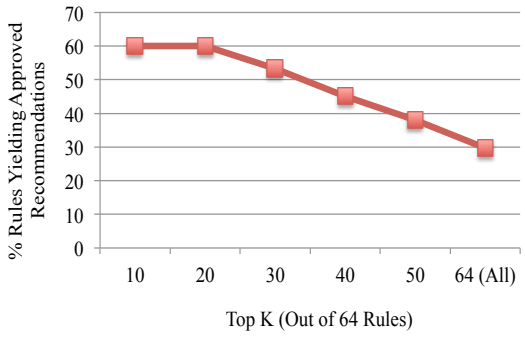


Figure 15: Percentage of Rules Yielding Approved Recommendations.

Section 3.1 since this case is not supported by existing association rule mining techniques. In the experiment, we set the confidence and support thresholds to values 95%, and 0.2%, respectively. In the x-axis of the figure, we vary the number of newly added annotations (δ) between 200 and 1,600. This is performed by isolating $delta$ annotations (publications) from the original dataset, discovering the association rules on the modified dataset, and then adding the isolated annotations back as the new delta batch. We perform the experiment on three different dataset sizes, i.e., small (20% of the entire dataset), medium (40% of the entire dataset), and large (the entire dataset). We compared between a naive non-incremental A-priori technique [8] versus the incremental technique proposed in Section 3.1. As the results show, the incremental algorithm outperforms the non-incremental one by up to two orders of magnitude while producing the same exact results.

In Figures 14 and 15, we investigate the virtue of the exploitation process within which the system proactively uses the discovered association rules and provides recommendations to enrich the annotated database. In the experiment, we set the confidence and support thresholds to 95% and 0.2%, respectively (resulting in 64 discovered rules as illustrated in Figure 11(b)). We then, in Figures 14 and 15 select the top K strongest rules for generating recommendations. The K varies over the values between 10 to 64 as indicated on the x-axis. Each rule, may generate many recommendations, e.g., attaching a specific annotations to some data tuples, and each recommendation is supported by some evidences (Refer to Figure 10). The results presented in Figure 14 and 15 illustrate for each K, the number, and percentage, respectively, of the rules that yield to at least one recommendation being approved by the database admin. This means that these rules were valuable as they

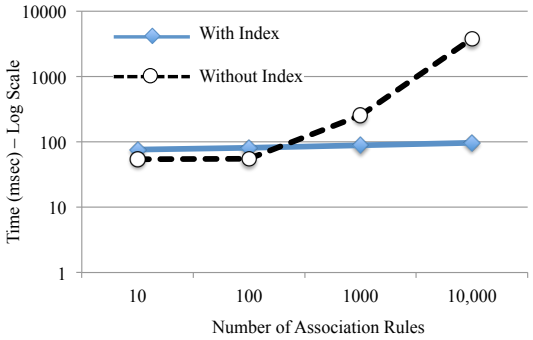


Figure 16: Searching for Matching Rules under New Tuples Insertions.

helped discovering missing attachments. The results show that a fair number of rules have yielded to approved recommendations. Moreover, the strongest rules, e.g., Top 10 or 20, usually yield to more realistic recommendations compared to the weaker rules, e.g., Top 50, or 60. The results in Figures 14 and 15 are interesting as they show that we may not even use all the discovered interesting rules for recommendation purposes. We can depend on only the top ranked ones to achieve high acceptance rate while reducing the domain experts' efforts in the verification process.

In Figure 16, we study the performance of searching for matching association rules under the insertion of new data tuples. We compare the two cases where the search does not use an index, i.e., scanning all existing rules, versus the use of index, i.e., the rules

are normalized and indexed using the B-Tree index (Refer to Section 4). In the experiment, we varied the number of rules from 10 to 10,000 (the x axis), and measured the search time (the y axis). The experiment is repeated 10 times, and the average values are presented in the figure. As expected, the *"With Index"* case scales much better and remains stable as the number of association rules gets larger. Whereas, the *"Without Index"* case has slightly better performance when the number of rules is very small. This is because the key lookups over the index—which are multiple lookups per tuple—add unnecessary overhead when the number of rules is very small. In this case, one scan over all rules becomes faster.

7. CONCLUSION

In this paper, we investigated a new facet of annotation management, which is the discovery and exploitation of the hidden annotation-related correlations. The addressed problem is driven by the emerging real-world applications that create and maintain large-scale repositories of annotated databases. The proposed work opens a new application domain to which the well-known association rule mining can be applied. We show cased several scenarios specific to annotated databases that cannot be efficiently handled by the state-of-art in association rule mining. We then proposed algorithms for efficient and incremental maintenance of the discovered association rules under these scenarios. We proposed two important applications for leveraging the discovered annotation-related correlations and enhancing the quality of the underlying database, which are the discovery of missing attachments, and the recommendation of applicable annotations to newly inserted data. To enable seamless use by scientists, we integrated the proposed algorithms within the annotation management engine and developed an end-to-end system including an Excel-based GUI through which all of the proposed functionalities can be performed.

8. REFERENCES

[1] eBird Trail Tracker Puts Millions of Eyes on the Sky. *https://www.fws.gov/refuges/RefugeUpdate/MayJune_2011/ebirdtrailtracker.html*.

[2] EcoliHouse: A Publicly Queryable Warehouse of E. coli K12 Databases. http://www.porteco.org/.

[3] Gene Ontology Consortium. http://geneontology.org.

[4] Hydrologic Information System CUAHSI-HIS. (http://his.cuahsi.org.

[5] The Avian Knowledge Network (AKN). http://www.avianknowledge.net/.

[6] The Universal Protein Resource Databases (UniProt) . http://www.ebi.ac.uk/uniprot/.

[7] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TKDE*, 17(6):734–749, 2005.

[8] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB, pages 487–499, 1994.

[9] D. Bhagwat, L. Chiticariu, and W. Tan. An annotation management system for relational databases. In *VLDB*, pages 900–911, 2004.

[10] A. J. B. Brush, D. Bargeron, A. Gupta, and J. Cadiz. Robust Annotation Positioning in Digital Documents. In *Proceedings of the 2001 ACM Conference on Human Factors in Computing Systems (CHI)*, pages 285–292. ACM Press, 2001.

[11] P. Buneman, A. Chapman, and J. Cheney. Provenance management in curated databases. In *SIGMOD*, pages 539–550, 2006.

[12] P. Buneman, J. Cheney, W.-C. Tan, and S. Vansummeren. Curated databases. In *Proceedings of the 27th ACM symposium on Principles of database systems (PODS)*, pages 1–12, 2008.

[13] P. Buneman and et. al. On propagation of deletions and annotations through views. In *PODS*, pages 150–158, 2002.

[14] P. Buneman, S. Khanna, and W. Tan. Why and where: A characterization of data provenance. *Lec. Notes in Comp. Sci.*, 1973:316–333, 2001.

[15] P. Buneman, E. V. Kostylev, and S. Vansummeren. Annotations are relative. In *Proceedings of the 16th International Conference on Database Theory*, ICDT '13, pages 177–188, 2013.

[16] D. Cheung, J. Han, V. Ng, and C. Wong. Maintenance of discovered association rules in large databases: an incremental updating technique. In *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*, pages 106–114, Feb 1996.

[17] L. Chiticariu, W.-C. Tan, and G. Vijayvargiya. DBNotes: a post-it system for relational databases based on provenance. In *SIGMOD*, pages 942–944, 2005.

[18] P. R. Christopher D. Manning and H. Schutze. Book Chapter: Text classification and Naive Bayes, in Introduction to Information Retrieval. In *Cambridge University Press*, pages 253–287, 2008.

[19] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web (WWW)*, pages 271–280, 2007.

[20] S. B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *SIGMOD*, pages 1345–1350, 2008.

[21] M. Eltabakh, W. Aref, A. Elmagarmid, and M. Ouzzani. Supporting annotations on relations. In *EDBT*, pages 379–390, 2009.

[22] A. Gattani and et. al. Entity extraction, linking, classification, and tagging for social media: a wikipedia-based approach. *Proc. VLDB Endow.*, 6(11):1126–1137, 2013.

[23] W. Gatterbauer, M. Balazinska, N. Khoussainova, and D. Suciu. Believe it or not: adding belief annotations to databases. *Proc. VLDB Endow.*, 2(1):1–12, 2009.

[24] F. Geerts and et. al. Mondrian: Annotating and querying databases through colors and blocks. In *ICDE*, pages 82–93, 2006.

[25] F. Geerts, A. Kementsietsidis, and D. Milano. iMONDRIAN: a visual tool to annotate and query scientific databases. In *Proceedings of the 10th international conference on Advances in Database Technology (EDBT)*, pages 1168–1171, 2006.

[26] F. Geerts and J. Van Den Bussche. Relational completeness of query languages for annotated databases. In *Proceedings of the 11th international conference on Database Programming Languages (DBPL)*, pages 127–137, 2007.

[27] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber. Scientific data management in the coming decade. *SIGMOD Record*, 34(4):34–41, 2005.

[28] T. J. Green. Containment of conjunctive queries on annotated relations. In *ICDT*, pages 296–309, 2009.

[29] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *In Proc. 1995 Int. Conf. Very Large Data Bases*, pages 420–431, 1995.

[30] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD, pages 1–12, 2000.

[31] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explor. Newsl.*, 2(1):58–64, June 2000.

[32] K. Ibrahim, D. Xiao, and M. Y. Eltabakh. Elevating Annotation Summaries To First-Class Citizens In InsightNotes. In *EDBT Conference*, 2015.

[33] J. Kahan and M.-R. Koivunen. Annotea: an open RDF infrastructure for shared Web annotations. In *Proceedings of the 10th international conference on World Wide Web (WWW)*, pages 623–632, 2001.

[34] E. V. Kostylev and P. Buneman. Combining dependent annotations for relational algebra. In *Proceedings of the 15th International Conference on Database Theory (ICDT)*, pages 196–207, 2012.

[35] L. S. P. Lee, Woei-Jyh; Raschid. Mining Meaningful Associations from Annotations in Life Science Data Resources. *Proceedings of the Conference on Data Integration for the Life Sciences*, 1(1), 2008.

[36] Q. Li, A. Labrinidis, and P. K. Chrysanthis. ViP: A User-Centric View-Based Annotation Framework for Scientific Data. In *Proceedings of the 20th international conference on Scientific and Statistical Database Management (SSDBM)*, pages 295–312, 2008.

[37] M. Markovic, P. Edwards, D. Corsar, and J. Z. Pan. DEMO: Managing the Provenance of Crowdsourced Disruption Reports. In *IPAW*, pages 209–213, 2012.

[38] C. MARSHALL. Annotation: from paper books to the digital library. ACM Digital Libraries (1997).

[39] C. MARSHALL. The future of Annotation in a Digital (paper) world. The 35th Annual GSLIS Clinic: Successes and Failures of Digital Libraries University of Illinois at Urbana-Champaign (1998).

[40] L. MartŠnez-Cruz, A. Rubio, M. MartŠnez-Chantar, A. Labarga, L. Barrio, and A. Podhorski. GARBAN: genomic analysis and rapid biological annotation of cDNA microarray and proteomic data. *Bioinformatics*, 19(16):2158–2163, 2003.

[41] B. Nath, D. K. Bhattacharyya, and A. Ghosh. Incremental association rule mining: a survey. *Wiley Interdisciplinary, Data Mining and Knowledge Discovery*, 3(3):157–169, 2013.

[42] A. Rae, B. Sigurbjörnsson, and R. van Zwol. Improving tag recommendation using social networks. In *Adaptivity, Personalization and Fusion of Heterogeneous Information*, RIAO, pages 92–99, 2010.

[43] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, 2005.

[44] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proceedings of the 21th International Conference on Very Large Data Bases*, VLDB, pages 407–419, 1995.

[45] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 1–12, 1996.

[46] W.-C. Tan. Containment of relational queries with annotation propagation. In *DBPL*, 2003.

[47] D. Tarboton, J. Horsburgh, and D. Maidment. CUAHSI Community Observations Data Model (ODM),Version 1.1, Design Specifications. In *Design Document*, 2008.

[48] P. TUCKER and D. JONES. Document annotation - to write, type or speak. In *International Journal of Man-Machine Studies*, pages 885–900, 1993.

[49] M. Twombly. Science online survey: Support for data curation. *Science Journal*, 331, 2011.

[50] D. Xiao and M. Y. Eltabakh. InsightNotes: Summary-Based Annotation Management in Relational Databases. In *SIGMOD Conference*, pages 661–672, 2014.

[51] H. Yang, D. T. Michaelides, C. Charlton, W. J. Browne, and L. Moreau. DEEP: A Provenance-Aware Executable Document System. In *IPAW'*, pages 24–38, 2012.

[52] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.