Mining for Patterns in Contradictory Data

Heiko Müller, Ulf Leser, Johann-Christoph Freytag

Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin, Germany

{hmueller, leser, freytag}@informatik.hu-berlin.de

ABSTRACT

Information integration is often faced with the problem that different data sources represent the same set of the real-world objects, but give conflicting values for specific properties of these objects. Within this paper we present a model of such conflicts and describe an algorithm for efficiently detecting patterns of conflicts in a pair of overlapping data sources. The contradiction patterns we can find are a special kind of association rules, describing regularities in conflicts occurring together with certain attribute values, pairs of attribute values, or with other conflicts. Therefore, we adapt existing association rule mining algorithms for mining contradiction patterns. Such patterns are an important tool for human experts that try to find and resolve problems in data quality using domain knowledge. We present the results of applying our method on a real world data set from the life science domain and show how it helps to generate clean data for integrated data warehouses.

1. INTRODUCTION

Collecting, manipulating, and analyzing data has become simple and valuable for many companies and organizations. Thus, a huge number of data collections exist. There are various scenarios where overlapping data collections are maintained, i.e., data sources that overlap in the set of real-world objects or facts represented. Overlapping data sources may be maintained in a controlled manner, such as replication of important data on different machines for load balancing or at different sites for security reasons, or uncontrolled, such as data being copied from websites or data being produced independently at different locations. Examples for uncontrolled overlaps are customer data collected by competing companies, census data managed by different governmental organizations, or scientific data produced and managed by different research groups. In this paper, we concentrate on the latter case.

Whenever data is distributed or generated without a control scheme enforcing consistency, there is a high probability that actual values will differ in the different data sets. Reasons can be processes on replicated data, modifying, filtering, or transforming the original values, errors in the replication mechanism, different levels of actuality of the data, or imprecision of measurement. Note that differences are not always errors; instead, an original might have contained errors which are corrected in one of the replicas.

A system trying to integrate such data sets faces several problems. First, it must identify inconsistencies in an efficient manner. Second, it must resolve these inconsistencies, e.g. by selecting a certain value due to a quality score of the data sources. This is usually pursued by scoring the sources according to data quality criteria. Data quality may be assessed using different quality criteria (dimensions), such as accuracy, completeness, uniqueness, or uniformity (representational consistency) [1][2]. Alternatively, resolution functions can be applied which combine the conflicting values into an "average" value (this could be a simple concatenation for conflicting character values). Third, the system will be interested in finding the source of the deviations to avoid such problems in future.

In this paper we provide a method to aid the developer of integrated systems in the first and the third task. We describe an algorithm for comparing pairs of overlapping data sources. The algorithm specifically searches and finds interesting conflicts in this comparison, i.e., inconsistencies that occur in some sense systematically or that follow a certain pattern. We call those cases *contradiction patterns*. Contradiction patterns are a very quick way to find quality hotspots in two data sets, since they help to ignore spurious problems. On the other hand, these patterns give to a human expert having the necessary domain knowledge valuable clues to possible reasons for data inconsistencies.

The method we present was developed in the course of the COLUMBA project which aims at integrating data on protein structures into a single, uniform data warehouse [3]. Data replication is very common in the life sciences. For example, the three databases GenBank [4], EMBL Nucleotide Sequence Database [5], and DNA Data Bank of Japan (DDBJ) [6] forming the *International Nucleotide Sequence Collaboration* claim to manage identical sets of information about DNA sequences, which is secured by nightly data exchange. An example of data that is distributed and modified due to different policies are protein structures. The global repository for protein structures is the Protein Data Base (PDB) [7]. However, two data cleansing projects exist, the PDB uniformity project [8] and the Macromolecular Structure Database (E-MSD) [9] that work independently to improve various aspects of the data in the PDB.

IQIS 2004 Maison de la Chimie, Paris, France.

As a result, there are currently three overlapping relational databases managing protein structure information, each containing roughly the same set of protein structures, but with many deviations in various properties of these structures. In the COLUMBA project, if there are conflicts in the data, we need to decide which of the values we should take into our warehouse. This task is non trivial; already finding all conflicts requires the generation of a large set of SQL queries. However, we quickly found that the amount of errors is overwhelming, and that we therefore need to focus on the most interesting (= most annoying) patterns in the errors. Using the algorithm presented in this work, we achieved this focusing, which lead to the detection of various parser errors, different understanding of data in the two cleansing projects, and, especially, truly conflicting data.

The paper is structured as follows. In Section 2 we present related work regarding the comparison of different data sets as well as the resolution of conflicts in overlapping data sources. Section 3 gives an introduction into contradictions and characteristics in contradictions using an example. We present a formal description of patterns in contradicting data in Section 4. Section 5 describes the algorithm for mining for patterns in contradicting data. Section 6 gives our results and experiences on applying our algorithm on a real data set, i.e., protein structure databases. We conclude in Section 7 and give an outlook into our further plans regarding the presented topic.

2. RELATED WORK

The areas of related work to our approach are assessment of data quality (regarding accuracy and uniformity), resolution functions, statistical comparison of data sets, and association rule mining. The later will be discussed in Section 5, together with our own algorithm for mining patterns in contradictions.

Assessing scores for the quality of data sources is a complex task as data quality is not a well-defined concept that can be measured and tracked with a crisp set of numbers. Assessing data quality has many subjective aspects [10]. Usually, quality is measured in various dimensions, such as accuracy, uniformity, timeliness, reliability, or consistency. Our approach mainly detects problems in accuracy and representational consistency by finding differences in overlapping data sources. This does not necessarily mean that non-contradicting values are always correct and consistent. However, verification of correctness is beyond the scope of our work.

Conflict resolution requires the definition of appropriate resolution functions [1][11]. These functions are usually defined individually for each of the attributes of the overlapping sources. Our work does not deal with resolution functions in itself, but can give valuable clues to the definition of such functions. Finding patterns in data errors helps the domain expert to concentrate on the most promising resolution functions covering the largest fraction of inconsistencies. Furthermore, specialized functions can be developed including the instance level of the data once patterns on this level are detected. However, conflict resolution must still fall back to general, i.e., instance-independent, resolution functions for those contradictions which are not covered by any of our patterns. In some cases the specification of a particular resolution function might also be impossible for a pattern because of the irregularity within the values (see Section 6).

The comparison of data sets using statistical methods is described in [10]. Other than with our approach the authors do not compare overlapping data sources representing the same real-world objects. They are not interested in the actual values causing the differences between the data sets. Using set comparison techniques they try to identify trends or other noticeable problems. For example, comparing customer data from different branches can reveal customer preferences and behavior by region, by group, or by month.

3. CONTRADICTIONS IN OVERLAPPING DATA SOURCES

Using two relation instances (short relations) r_1 and r_2 as shown in Figure 1, we illustrate contradicting data and characteristics in contradicting data. Both relations follow the simple schema

EXPERIMENT(ID, VOLUME, TEMP, METHOD, RESULT)

managing information about identical experiments performed by two laboratories or scientists both storing their results in individual relations. Each experiment has a unique identifier. Also recorded are the experimental method used, the volume of the sample, the temperature the experiment was performed at, and the result obtained. The primary key attribute ID allows a simple identification of identical experiments across the two relations. We are aware that this is a strong simplification, as object identification remains a major challenge in many overlapping real world data sets. However, in this paper we focus on finding mismatch pattern and deliberately ignore the problem of duplicate detection.

r_1

ID	VOL.	TEMP.	METHOD	RESULT
1	5.2	17	X-RAY	20.3
2	4.3	19	NMR	15.6
3	2.1	21	X-RAY	12.8
4	2.1	21	NMR	13.2
5	4.0	15	NMR	15.0

r_2

ID	VOL.	TEMP.	METHOD	RESULT
1	5.2	63	X-RAY	25
2	4.3	66	NMR	15.6
3	2.1	70	X-RAY	18
4	2.1	70	NMR	13.2
5	40	59	NMR	15.0
6	3.4	68	NMR	17.2

Figure 1. Data about identical experiments from different laboratories or scientists. Contradicting values are highlighted

Both relations differ in the number of experiments performed as well as in the attribute values. In this paper we do not consider differences concerning the number of objects represented in each of the relations, i.e., we do not assess the completeness of the individual relations. Instead, we only consider the subset of experiments performed by both laboratories. Tuples with matching primary key values from each of the relations are called *matching pairs*. *Contradictions* or conflicts occur within matching pairs when they have different values for the same attribute. The shaded cells in the relations of Figure 1 depict the contradicting values.

3.1 Reasons for Contradictions

There may be several reasons for the existence of contradictions in overlapping data, affecting different data quality criteria [12]. Common examples are:

- *Different data production strategies (workflows)*: Different proceeding in data collection and analysis often leads to different values for a certain property. This affects accuracy of the integrated data because one of the values is probably incorrect.
- *Different value representations*: Different sources may represent the same fact using different values, for example because they use different measure units. One example is the representation of an employees salary using $ in one source and € in another leading to different values for the same fact (the persons salary). This affects uniformity of the values between the sources, but not necessarily within each of the sources. Still, when integrating the sources we have to decide which of the representations to use.
- *Transformations or manipulations*: In the case of replicated relations, transformations or manipulations performed on one source may remain unnoticed by the other source. In other cases we take an existing relation as basis and perform different transformation sequences and receive different relations as result. Depending on the transformations performed, the resulting relations either differ in their accuracy or their uniformity. They might also differ in their completeness, i.e., tuples have been added or deleted, and their timeliness, i.e., missed updates result in relations contain data of different timeliness.
- *Systematic errors*: Systematic errors are introduced during data collection or analysis, for example by equipment malfunction. Systematic errors result in inaccuracies. They might also be connected to special conditions on the values in other attributes. For example, as described below, when using different experimental methods one laboratory might have a malfunctioning equipment which leads to differences in the result values each time this particular experimental method is used.
- *Arbitrary errors*: Arbitrary errors (noise) may be introduced by chance while collecting, manipulating or analyzing data. Examples include human errors or imprecision of measurements. They also lead to inaccuracies.

For conflict resolution we aim to identify characteristics in the values regarding the occurrence of contradictions within each of the attributes (except the primary key attribute). We call these characteristics *patterns in contradicting data* (short *contradiction pattern*). These patterns help in providing answers to questions like "Which are the conflict-causing attributes, values, or value pairs?" and "What kind of dependencies exists between the occurrence of contradictions in different attributes?". A pattern is therefore a characteristic combination of values occurring with a certain frequency in conjunction with contradictions in a certain attribute. For a pattern to be of interest regarding the explanation of the contradictions it should not occur in combination with matching pairs that do not have a conflict in this exact attribute. Using the identified patterns, a domain expert can identify reasons for contradictions and specify proper actions for conflict resolution.

To highlight the usefulness of this concept we give examples. Consider again Figure 1. Obviously, there is a contradiction in each matching pair within attribute **Temperature**. A good guess is that this high contradiction frequency results from different representations used for experimental temperature in both relations. In the first relation Celsius is used as temperature scale while the second relation uses Fahrenheit. For further assessment of value accuracy we can transform one of the representations into the other and again determine existing contradictions (inaccuracies). Regarding the characteristics concerning the conflicts in attribute **Result**, some inspection shows that, every time a contradiction occurs within this attribute, the experimental method used by both sources is 'X-RAY'. The domain expert might conclude that one of the laboratories or scientist uses erroneous equipment or makes errors in determine the result using the accordant equipment. On the other hand, the contradiction in attribute **Volume** in experiment 5 appears to result from an arbitrary error, in this case probably a typographic error. Rare contradiction patterns like this one are likely to be left out by the pattern mining algorithm, depending on the parameter values used as described below.

4. CONTRADICTION PATTERNS

In this Section we give a formal definition of contradiction patterns between two relations containing an overlapping set of real-world objects. As mentioned above we consider two relation instances r_1 and r_2 following the relation schema $R(\underline{ID}, A_1, \ldots, A_n)$ with ID being the primary key. Values for the individual attributes for each of the tuples in the relations are denoted by $t[A_i]$. The set of matching pairs M is defined as

$$M = \{(t_1, t_2) \in r_1 \times r_2 \mid t_1[ID] = t_2[ID]\}$$

i.e., a set of tuple pairs, one tuple from each relation, having equal values in their primary key and therefore representing the same real-world object. We use $m_i = (t_1, t_2)$ as an abbreviation for the i^{th} element of M with $m_i[1] = t_1$ and $m_i[2] = t_2$.

The Boolean function *conflict*(m, A_i) indicates whether contradicting values exists in the matching pair m for attribute A_i.

$$\textit{conflict}(m, A_i) = \begin{cases} \textit{true}, \text{ if } m[1][A_i] \neq m[2][A_i] \\ \textit{false}, \text{ else} \end{cases}$$

For each of the attributes $A_1, \ldots, A_n$ we define C_{Ai} as the subset of matching pairs having contradicting values for attribute A_i

$$C_{Ai} \subseteq M = \{m \in M \mid \textit{conflict}(m, A_i) = \textit{true}\}$$

As the complement, we define N_{Ai} as the subset of matching pairs not possessing a contradiction in attribute A_i

$$N_{Ai} \subseteq M = \{m \mid \textit{conflict}(m, A_i) = \textit{false}\}$$

Clearly, it holds that $C_{Ai} \cup N_{Ai} = M$ and $C_{Ai} \cap N_{Ai} = \emptyset$.

Furthermore, we shall later use the conflict frequency for attribute A_i defined as

$$cf_{Ai} = \frac{|\ C_{Ai}\ |}{|\ M\ |}$$

We now have all the tools together to define what a contradiction pattern is. First, we define what a pattern is. A pattern ρ is a Boolean function over the matching pairs describing a specific data characteristic regarding the elements of M. We consider patterns that are conjunctions of terms, where each term has one of the two following forms:

$$m[j][A_i] = x \text{ or } \mathit{conflict}(m, A_i) = \mathit{true} \mid \mathit{false}$$

where $1 \le j \le 2$ and $1 \le i \le n$. A matching pair m satisfies ρ if $\rho(m) = \mathit{true}$, and violates ρ if $\rho(m) = \mathit{false}$.

Recall the occurrence of contradictions in the attribute **Result** in the example above. These contradictions can be described by the following pattern:

$$\rho = m[1][\text{METHOD}] = \text{'X-RAY'} \wedge \mathit{conflict}(m, \text{RESULT}) = \mathit{true}$$

However, one is usually not only interested in patterns holding for a single matching pair. Instead, we are searching patterns holding for more than one pair. To define such interesting patterns, we need some more definitions. Note that we shall describe the relationships to the classical definitions of support and confidence in the next section.

The support of ρ in set M, denoted by *support*(ρ, M), gives the relative frequency of matching pairs from M satisfying ρ in the total number of elements in M:

$$\mathit{support}(\rho, M) = \frac{|\ \{m \in M \mid \rho(m) = \mathit{true}\}\ |}{|\ M\ |}$$

A pattern ρ is called a contradiction pattern for an attribute A_i, denoted by ρ_{Ai}, if it has a support above a threshold *min-inside-support* within the subset of contradicting pairs C_{Ai},

$$\mathit{support}(\rho_{Ai}, C_{Ai}) \ge \mathit{min\text{-}inside\text{-}support},$$

and a support below threshold *max-outside-support* in the subset of non-contradicting matching pairs N_{Ai},

$$\mathit{support}(\rho_{Ai}, N_{Ai}) \le \mathit{max\text{-}outside\text{-}support},$$

i.e., the pattern ρ_{Ai} describes a data characteristic occurring frequently in matching pairs having contradictions in attribute A_i and rarely in matching pairs not having a contradiction in attribute A_i. Let $P(A_i)$ denote the set of all contradiction patterns for attribute A_i for which these conditions hold.

4.1 Contradiction Patterns as Association Rules

A contradiction pattern ρ_{Ai} can be seen as a special association rule $A \leftrightarrow B$ with A being a term of type $\mathit{conflict}(m, A_i)$ and B a contradiction pattern ρ_{Ai}. The contradiction pattern B has to have a support above *min-inside-support* for C_{Ai} which is equal to $\mathit{confidence}(A \rightarrow B) \ge \mathit{min\text{-}inside\text{-}support}$ with

$$\mathit{confidence}(A \rightarrow B) = \frac{|\ \{m \in M \mid \mathit{conflict}(m, A_i) \wedge B(m)\}\ |}{|\ \{m \in M \mid \mathit{conflict}(m, A_i)\}\ |}$$

We therefore call *min-inside-support* the *conflict relevance*, i.e., the number of elements in C_{Ai} satisfying B. The contradiction pattern B is also not allowed to have support in N_{Ai} above *max-outside-support* which is equal to $\mathit{confidence}(B \rightarrow A) \le 1 - \mathit{max\text{-}outside\text{-}support}$ with

$$\mathit{confidence}(B \rightarrow A) = \frac{|\ \{m \in M \mid \mathit{conflict}(m, A_i) \wedge B(m)\}\ |}{|\ \{m \in M \mid B(m)\}\ |}$$

We define *conflict potential* as *1 – max-outside-support*, i.e., the probability (or potential) that a matching pair satisfying pattern B has a contradiction in attribute A_i.

It is important to note that for a contradiction pattern ρ_{Ai} we only want to include terms T_i that are relevant or interesting in conjunction with the occurrence of a certain type of contradiction in the attribute A_i. This means that we do not want to combine terms with largely differing conflict relevancies, as their combination might not yield any important information for the domain expert evaluating the computed contraction patterns afterwards. Therefore, we use a *support-deviation* threshold. Let $T_1,\ldots,T_q$ be the set of terms contained in a contradiction pattern ρ_{Ai}. Then we require the following condition to hold ρ_{Ai}:

$$\mathit{support\text{-}deviation} \le \frac{\mathit{min}(\mathit{support}(T_i, C_{Ai}))}{\mathit{max}(\mathit{support}(T_i, C_{Ai}))},\ 1 \le i \le q$$

Following the definition in [13] we call this the *contradiction-cross-support-property*, which is useful for avoiding patterns containing terms with widely differing support levels.

5. MINING FOR CONTRADICTION PATTERNS

In this Section we describe the algorithm developed by us for mining contradiction patterns as defined above. We start by discussing why the common data mining algorithms as first described in [14] are not applicable to our problem.

5.1 Common Frequent Itemset Mining

Contradiction patterns are special cases of association rules. This suggests the usage of existing association rule mining algorithms. However, in contrast to the common basket data example, we do not solely have binary attributes in our relations, but our attributes are primarily quantitative. This problem can be bypassed by transforming for each attribute A_i each occurring value x into a binary attribute named "A_i=x" of a resulting relation r_{bin}. Within r_{bin} each tuple t' has $t'[A_i{=}x] = 1$ if the according tuple $t \in r$ has value x in attribute A_i, i.e., $t[A_i] = x$, or 0 if $t[A_i] \ne x$. Figure 2 shows a section of the resulting relation r_{bin1} for transforming relation r_1 from the example above.

After transforming relations r_1 and r_2 into relations r_{bin1} and r_{bin2}, we can define M, denoted as the *view of matching pairs*, as a view over relations r_{bin1} and r_{bin2} with each matching pair m becoming one tuple in the resulting view. Within this view we also introduce for each attribute A_i from R a *contradiction indicator* $CI[A_i]$ with

$$CI[A_i] = \begin{cases} 1, \text{ if } \mathit{conflict}(m, A_i) = \mathit{true} \\ 0, \text{ else} \end{cases}$$

As a result we receive a view M containing only binary attributes. This approach has a main disadvantage: For attributes containing many different values the transformation leads to relations with a large amount of attributes containing only sparse data, i.e., very few "1" values. Introducing the contradiction indicator $CI[A_i]$ results in dense attributes for those A_i having a high contradiction frequency cf_{Ai}. Using the common support threshold to prune the combinatorial search space when applying association rule mining algorithms on these relations poses the problems as outlined in [15][13]: (i) if the minimum support threshold is too low, many uninteresting patterns involving items with substantially different support levels are extracted, and (ii) if the minimum support threshold is too high, many strong affinity pattern involving items with low support levels are missed. Using intervals instead of quantitative attribute values as described in [15] solves this dilemma. However, for our problem, value intervals are unsatisfactory because we are interested in the actual values causing the mismatches.

r_{bin1}

ID	…	METHOD=X-RAY	METHOD=NMR	…
1	…	1	0	…
2	…	0	1	…
3	…	1	0	…
4	…	0	1	…
5	…	0	1	…

Figure 2. Transformation of relation r_1 into a binary relation r_{bin1}, exemplified for attribute METHOD

5.2 View of Matching Pairs

We decided to use the quantitative attributes within the view of matching pairs. In SQL, our view M is defined as

```
CREATE VIEW M AS
SELECT
  r1.ID AS ID,
  r1.A1 AS r1[A1],
  r2.A1 AS r2[A1],
  CASE r1.A1 = r2.A1 THEN 0 ELSE 1 AS CI[A1],
  ...
FROM r1, r2
WHERE r1.ID = r2.ID
```

The resulting view has the following schema:

$$M\ (\underline{ID},\ r_1.A_1,\ r_2.A_1,\ CI[A_1],\ \ldots,\ r_1.A_n,\ r_2.A_n,\ CI[A_n])$$

The subsets C_{Ai} can now be expressed as views on M:

```
CREATE VIEW CAi AS
SELECT * FROM M WHERE CI[Ai] = 1
```

Figure 3 shows a fraction of the resulting view of matching pairs for attribute **Result**. The subset C_{RESULT} of matching pairs showing a contradiction for attribute **Result** is shaded.

M

ID	…	CI[RESULT]	R_1[RESULT]	R_2[RESULT]	…
1	..	1	20.3	25	.
2	…	0	15.6	15.6	…
3	..	1	12.8	18	.
4	…	0	13.2	13.2	…
5	…	0	15.0	15.0	…

Figure 3. Fraction of view of matching pairs for attribute Result with C_{RESULT} being highlighted

5.3 Pattern Mining Algorithm

The algorithm for contradiction pattern mining is depicted in Figure 4. It determines for each attribute $A_i \in R$ the set of contradiction patterns $P(A_i)$. As we are not using solely binary attributes our items have the form of the pattern terms as defined in Section 4. The algorithm uses the three parameters *conflict relevance*, *conflict potential*, and *support-deviation* and is composed of two main parts. In the first part we determine the initial set of candidate patterns for each attribute A_i, i.e., terms $r_1.A_j = x$, $r_2.A_j = x$, and $CI[A_k] = [0, 1]$ with $1 \le j \le n$ and $1 \le k \le n \wedge k \ne i$, having support in C_{Ai} above the given threshold *conflict relevance* and a conflict potential above the specified *conflict potential* threshold.

cr: conflict relevance; cp: conflict potential, sd: support deviation

```
FOR EACH CI[Ai] ∈ M
 P(Ai) := ∅; CandAi := ∅
 -- First part
  FOR EACH Bi ∈ M, Bi ≠ CI[Ai]
   FOR EACH T HAVING support(T, CAi) ≥ cr
    if (confidence(T → CI[Ai]) ≥ cp)
     CandAi := CandAi ∪ {ρAi = (Ai, T)}
   END FOR
  END FOR
 -- Second part
  WHILE CandAi ≠ ∅
   NewCandAi := ∅
   FOR EACH ρi ∈ CandAi
    FOR EACH ρj ∈ CandAi; ρi ≠ ρj
     if (support_deviation(ρi, ρj) ≥ sd)
      ρm := merge(ρi, ρj)
      if (support(ρm, CAi) ≥ cr)
       NewCandAi := NewCandAi ∪ {ρm}
    END FOR
    if (NOT was_merged(ρi))
     P(Ai) := P(Ai) ∪ ρI
   CandAi := NewCandAi
   END FOR
  END WHILE
END FOR
```

Figure 4. Conflict Pattern Mining Algorithm

The first part can be implemented using two types of aggregation queries against the view of matching pairs. For the determination of terms having sufficient support within C_{Ai} the query is

```
SELECT R1[Ai], COUNT(CI[Ai])
FROM M
WHERE CI[Ai] = 1
GROUP BY R1[Ai]
HAVING COUNT(CI[Ai]) > min-tuple-count
```

with

$$min\text{-}tuple\text{-}count = \frac{conflict\ relevance * 100}{cf_{Ai} * |M|}$$

For calculation of the confidence values $confidence(T \rightarrow CI[A_i])$ we execute for each of the terms resulting from the first query a second query

```
SELECT COUNT(*) FROM M WHERE R1[Ai] = x
```

In the second part of the algorithm we successively extend the number of terms of the candidate patterns by merging them. We thereby adhere the contradiction-cross-support-property by only merging patterns if their terms have a support deviation within the given threshold. If the merged pattern still has support within C_{Ai} above the conflict relevance threshold it is used as candidate pattern in the next iteration. Candidate pattern, which could not be further extended, are added to $P(A_i)$. With this approach $P(A_i)$ contains only maximized patterns, i.e., contradiction patterns not being a subset of another pattern from $P(A_i)$. The second part of the algorithm resembles the application of the Hyperclique Miner Algorithm [13] to the subset of contradiction matching pairs C_{Ai}, with our support deviation threshold being analogous to the h-confidence.

6. EXPERIMENTAL RESULTS AND REMAINING PROBLEMS

We implemented the algorithm described using the Java™ Programming Language and IBM DB2™ as the relational database management system. The experimental data was protein structure data from PDB. We used two different versions of the data which are available from the COLUMBA-project [3]: (i) the original PDB data available in flat file format, denoted as PDB, and (ii) data in macromolecular Crystallographic Information File format (mmCIF) from the PDB uniformity project at the University of California San Diego (UCSD), denoted as MMS. We currently utilize the BioPython parser for PDB flat-files and the OpenMMS Toolkit, containing software for parsing and loading mmCIF files into a relational database.

6.1 Experimental Results

From the described data sources we used the relation PDB_ENTRY containing information about entries in PDB like the entry name, the entry deposition date and year, the resolution of the protein structure described as well as the experimental method used for resolution determination. Identification of matching tuples is trivial since both sources use the original PDB_ID for all entries in this table.

Table 1 shows the attributes of the relation PDB_ENTRY and some statistic information about (i) the number of contradictions within the attribute, (ii) the number of distinct values in PDB ∪ MMS, (iii) the number of distinct values in PDB, and (iv) the number of distinct values in MMS. Relation PDB_ENTRY has 23,649 tuples in source PDB and 24,200 in source MMS. The number of matching pairs is 23,570.

Table 1. Statistics about the attribute and their values in the view of matching pairs resulting from PDB and OPENMMS.

Attribute	$\|C_{Ai}\|$	# Values	# Values PDB	# Values MMS
NAME	23,570	39,669	19,056	20,613
YEAR_DEPOSIT	72	34	32	32
DEPOSIT_DATE	118	4,048	4,012	4,041
RELEASE_DATE	11,024	1,592	1,469	1,249
METHOD	23,520	111	46	87
RESOLUTION	9,742	405	310	375
R_VALUE	23,558	1,016	1	1016
PROGRAM	23,570	416	1	415

Table 2 shows the number of generated contradiction patterns for each of the attributes using different thresholds for *conflict relevance* and *conflict potential*. The *support deviation* is 95% in all of these experiments.

Table 2. Experimental results showing the number of contradiction patterns per attribute for different parameter values (*conflict relevance* / *conflict potential*).

Attribute	2% / 95 %	25% / 95%	2% / 50%	25% / 50%
NAME	136	66	400	290
YEAR_DEPOSIT	8	0	9	1
DEPOSIT_DATE	7	1	12	1
RELEASE_DATE	0	0	19	0
METHOD	136	66	384	274
RESOLUTION	17	3	34	4
R_VALUE	136	66	400	290
PROGRAM	136	66	400	290
Total Number	**576**	**268**	**1,658**	**1,150**

In the following we show some of the contradiction patterns generated by our various experiments and try to give an interpretation for them[1]:

$$\rho_{YEAR\_DEPOSIT} : MMS[YEAR\_DEPOSIT] = \texttt{'1900'}$$

[1] We use $MMS[A_i] = x$ and $PDB[A_i] = x$ as short form for the appropriate $m[j][A_i] = x$ terms

This pattern has a conflict relevance of about 50%. Each time the value `'1900'` occurs in attribute YEAR_DEPOSIT in MMS the according value in PDB is `'2000'`. This probably results from a parsing error as the values in the original PDB flat-files are represented in format DD-MMM-YY. This gives an important hint to a possible resolution of the problem. Another example is the pattern

$$\rho_{\text{DEPOSIT\_DATE}} : \text{CI[YEAR\_DEPOSIT]} = 1$$

This follows directly as YEAR_DEPOSIT is the year fraction of DEPOSIT_DATE. Having determined this, we can follow that the conflicts in YEAR_DEPOSIT can be ignored since the problem must be resolved in the DEPOSIT_DATE. A particularly interesting pattern is the following:

$$\rho_{\text{RESOLUTION}} : \text{MMS[METHOD]} = \texttt{'NMR'}$$

When taking a closer look at the values involved in attributes RESOLUTION and METHOD we see that neither sources stores meaningful values for RESOLUTION if METHOD is `'NMR'`. However, the sources represent this fact differently: PDB uses the value `0.0`, while MMS uses a `NULL`. This explains the contradiction.

However, not all contradiction patterns the algorithm reports make sense. Consider the following two rules:

$$\rho_{\text{NAME}} : \text{CI[RELEASE\_DATE]} = 0$$

$$\rho_{\text{NAME}} : \text{CI[RELEASE\_DATE]} = 1$$

Both rules are found due to the reason that both terms CI[RELEASE_DATE] = 0 and CI[RELEASE_DATE] = 1 have high support within C_{NAME}. However, taken together they clearly do not provide helpful clues to a human expert.

6.2 Discussion and Remaining Problems

We have presented an algorithm for finding patterns of contradictions in semantically overlapping, yet different data sets. Whenever such data sets are to be merged into a uniform databases with "a single truth", conflicts need to be identified and resolved which usually requires costly expert inspection. Our algorithm helps in that it points the experts attention to the most prevalent patterns of mismatches.

However, there remain a couple of possible improvements. A particularly difficult problem is the selection of appropriate parameters for the parameter of the algorithm, i.e., the thresholds for conflict relevance and conflict potential. Choosing low values for *conflict relevance* identifies patterns containing rarely occurring values. For example, in MMS the value `'?'` occurs four times in attribute YEAR_DEPOSIT causing a conflict within the view of matching pairs. This information is found when using low *conflict relevance*. On the other hand, low *conflict relevance* has a drawback regarding attributes having high conflict frequencies resulting from differing values or heterogeneous representations within the sources being compared. One such case in our example is the NAME attribute. NAME has a high divergence in its values, which causes a conflict where each value occurs only a few times (as can be see in Table 1). Each of the contradicting value pairs is identified a contradiction pattern when using low *conflict relevance* resulting in a large amount of rather uninteresting patterns.

A second problem concerns a low *conflict relevance* threshold in the presence of attributes having a high contradiction frequency In the result we receive many patterns include values that appear several times within other attributes but normally do not have any relationship to a conflict cause (see the last example in Section 6.1). Ideally, we would find a strong correlation for them with terms using only the values of the attribute under concern. Another approach would be to relate the conflict relevance value with the conflict frequency of the attribute.

The choice of a higher *conflict potential* usually leads to larger number of identified contradiction patterns because values are included as pattern terms, which also occur outside the subset of conflicting matching pairs C_{Ai}. In some cases this still can yield interesting information as shown in the following example where the patterns returned do not contain all the relevant information concerning the cause for the contradiction. The pattern

$$\rho_{\text{YEAR\_ DEPOSIT}} : \text{MMS[YEAR\_ DEPOSIT]} = \texttt{'1900'}$$

does not reveal that PDB[YEAR_DEPOSIT] is always `'2000'` in these contradiction cases, because there are several matching pairs with year 2000 as deposition year outside of $C_{\text{YEAR\_DEPOSIT}}$. Finding such regularities requires manual inspection of the primary results or lower conflict potential values. Anyway, the algorithm clearly tells the user where it is worthwhile to look into the details.

Another problem remains with the number of candidate patterns generated and tested within the second part of the algorithm. The number is very large if there are many attributes with high conflict frequency, or many attributes with very few but frequent values. These attributes form very long patterns when the attribute under concern also has a high contradiction frequency, which hinders pruning using the method and parameters described. A possible solution would be to exclude attributes from the mining process, which obviously differ because of their value representations. These kind of attributes normally are not involved in meaningful contradiction patterns. This can be done within a preprocessing stage in which the domain expert evaluates statistics about the single attributes and flags those to be excluded.

7. CONCLUSIONS AND OUTLOOK

Within this paper we presented our idea of mining for characteristic patterns in contradicting data. Those patterns give valuable clues for subsequent steps for dealing with and improving data quality. We formally defined contradiction patterns, their properties, and their relation with association rules. By adapting existing association rule mining algorithms, we described an algorithm for contradiction pattern mining. The algorithm was implemented and we showed some preliminary results we obtained on a real-world data set taken from the domain of life science databases. We discussed some of the remaining problems with our algorithm and proposed possible solutions.

In this paper we focus solely on comparing two data sources. An intended extension of our method is the comparison of n data sources. Another planned future work concerns 1:n- and n:m-relationships between relations in a database. We plan on extending the algorithm to also mine contradiction patterns between the sets of related tuples.

8. ACKNOWLEDGMENTS

This work is supported by BMBF grant no. 0312705B (Berlin Center for Genome-Based Bioinformatics) and DFG grant number GRK 316 (Berlin-Brandenburg School on Distributed Information Systems). We thank Kristian Rother for implementing the parser

for PDB data and Felix Naumann for many comments on the project and this paper.

9. REFERENCES

[1] Naumann, F., *Quality-Driven Query Answering for Integrated Information Systems*. Lecture Notes in Computer Science, LNCS 2261, Springer, 2002

[2] Olson, J.E., *Data Quality – The Accuracy Dimension.* Morgan Kaufmann Publishers, ISBN1-55860-891-5, 2003

[3] Rother, K., Müller, H., Trissl, S., Koch, I., Steinke, T., Preissner, R., Frömmel, C., Leser, U., *COLUMBA: Multidimensional Data Integration of Protein Annotations.* International Workshop on Data Integration in Life Sciences (DILS 2004), Leipzig, Germany

[4] Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Wheeler, D.L. *GenBank.* Nucleic Acids Res., Vol. 31(1) (2003) 23-37

[5] Kulikova, T., Aldebert, P., Althorpe, N., Baker, W., Bates, K., Browne, P., van den Broek, A., Cochrane, G., Duggan, K., Eberhardt, R., Faruque, N., Garcia-Pastor, M., Harte, N., Kanz, C., Leinonen, R., Lin, Q., Lombard, V., Lopez, R., Mancuso, R., McHale, M., Nardone, F., Silventoinen, V., Stoehr, P., Stoesser, G., Tuli, M.A., Tzouvara, K., Vaughan, R., Wu, D., Zhu, W., Apweiler. R., *The EMBL Nucleotide Sequence Database* **.** Nucl. Acids. Res. 2004 32: D27-D30.

[6] Tateno, Y., Imanishi, T., Miyazaki, S., Fukami-Kobayashi, K., Saitou, N., Sugawara, H., Gojobori, T., *DNA Data Bank of Japan (DDBJ) for genome scale research in life science.* Nucl. Acids. Res. 2002 30: 27-30

[7] Bernstein, F.C., Koetzle, T.F., Williams, G.J.B., Meyer, E.F. Jr., Brice, M.D., Rodgers, J.R., Kennard, O., Shimanouchi, T., Tasumi, M., *The Protein Data Bank: a computer-based archival file for macromolecular structures.* J. Mol. Biol., Vol. 112 (1977) 535-542

[8] Bhat, T.N., Bourne, P., Feng, Z., Gilliland, G., Jain, S., Ravichandran, V., Schneider, B., Schneider, K., Thanki, N., Weissig, H., Westbrook, J., Berman, H.M., *The PDB data uniformity project*. Nucleic Acid Research, Vol. 29(1) (2001) 214-218

[9] Boutselakis, H., Dimitropoulos, D., Fillon, J., Golovin, A., Henrick, K., Hussain, A., Ionides, J., John, M., Keller, P.A., Krissinel, E., McNeil, P., Naim, A., Newman, R., Oldfield, T., Pineda, J., Rachedi, A., Copeland, J., Sitnov, A., Sobhany, S., Suarez-Urunea, A., Swaminathan, J., Tagari, M., Tate, J., Tromm, S., Velankar, S., Vranken, W.; *E-MSD: the European Bioinformatics Institute Macromolecular Structure Database*. Nucleic Acid Research, Vol. 31(1) (2003) 458-462

[10] Dasu, T., Johnson, T., *Exploratory Data Mining and Data Cleaning.* Wiley-Interscience, ISBN 0-471-26851-8, 2003

[11] Naumann, F., Haeussler, M., *Declarative Data Merging with Conflict Resolution*. Proceedings of the International Conference on Information Quality (IQ 2002), Cambridge, MA, 2002

[12] Müller, H., Naumann, F., Freytag, J.C., *Data Quality in Genome Databases*. Proceedings of the International Conference on Information Quality (IQ 2003), Cambridge, MA, 2003

[13] Hui Xiong, Pang-Ning Tan, Kumar, V., *Mining Strong Affinity Association Patterns in Data Sets with Skewed Support Distribution.* Third IEEE International Conference on Data Mining, Melbourne, Florida, November 19 - 22, 2003

[14] Agrawal, R., Srikant, R. *Fast Algorithms for Mining Association Rules.* Proc. Of the 20th Int'l Conf. On Very Large Data Bases, Santiago, Chile, Sept. 1994

[15] Srikant, R, Agrawal, R *Mining Quantitative Association Rules in Large Relational Tables. Proc. of the ACM-SIGMOD 1996 Conference on Management of Data*, Montreal, Canada, June 1996