Clean Answers over Dirty Databases: A Probabilistic Approach

DBMS Course Project Report

CS702: Database Engineering

Submitted By:
Bhaskar Gautam (16CS04F)
and
Nitin Kumar Yadav
(16CS10F)

Department of Computer Science and Engineering
National Institute of Technology, Karnataka

Contents

List of Figures

Abstract

Dirty data is incorrect or incomplete even inconsistent data which can be generated due to improper methods in data management and data storage. Cleaning and integration of data is the challenging task in such domain. Existing techniques can identify affected tuples, the merging or elimination of duplicates can be a difficult task that relies on ad-hoc and often manual solutions. Through paper [3], Authors propose a complementary approach that permits declarative query answering over duplicated data, where each duplicate is associated with a probability of being in the clean database and try to rewrite queries over a database containing duplicates to return each answer with the probability that the answer is in the clean database which are sensitive to the semantics of duplication and help a user understand which query answers are most likely to be present in the clean database.
The data cleaning and integration equivalent of authors is independent of the way the probabilities are produced, but is able to effectively exploit them during query answering but in the absence of external knowledge that associates each database tuple with a probability, author offer a technique, based on tuple summaries, that automates this task. Author studies show that the rewriting does not introduce a significant overhead in query execution time. Considering this paper [3] as a base project, I will extend the project further with different methodology in order to determine whether results can be improved from the state of art or not.

Chapter 1

Introduction

In data cleaning and integration, detection of duplicate tuples (occur due to data entry error or integration of different databases) is an important task. There are many techniques and tools exist for detecting duplicate tuples. These are Commercial Data Integration Tools based on clustering and statistical techniques, like IBM websphere, First logic Information Quality etc.
While integrating different databases by resolving structural and semantic differences, there may be duplication due to different information about same entity by different sources. In such cases, Commercial Integration Tools uses conflict resolution rules to merge the tuples of same entity, like averaging of income of same employee from different sources databases.
But these rules do not work on categorical data, where values cannot be added or combined, like employee X lives in city A, informed by source S1 and same employee X lives in city B informed by source S2. There are no such set of rules that can resolve all current and future conflicts completely and correctly. Therefore some Data Integration Tools keep Matching Tuples in Integrated Database. So we need some techniques to solve this problem of cleaning queries answers over dirty database. By using existing data integration tools, queries can be executed directly on dirty database but it fails to address duplicate semantics.

Table 1.1: Reference Sample Table

EMPID	INCOME	Source
E1	80	S1
E2	120	S2

For query to find employee IDs whose income is greater \$100k, answer will be E2. But problem is that we do not know whether it is correct or not with high certain.As there are more duplicate tuples which are showing different representations of same entity, in Clean database there will be only one representation for each entity. But absence of all correct conflict resolution rules, we include

these values from duplicates in clean database and adopt Probabilistic Approach for querying dirty database.
We need to associate probability to each of the tuples in inconsistent database.

Table 1.2: A dirty customer database: Loyalty Card

cardId	custFk	prob
111	c1	0.4
111	c2	0.6

Table 1.3: A dirty customer database: customer

name	income	prob
John	120	0.9
John	80	0.1
Mary	140	0.4
Marion	40	0.6

This probability assignment can be depend on source reliability or uniform. These probabilities will be output of tupple matching process of data integration tool. In these probabilistic query answering, a set of answers is computed together with their probability of being correct.

Real-world data is commonly integrated from multiple sources, and the integration process may lead to a variety of data errors, such as incorrect values and duplicate representations of the same real-world entity. Therefore, even running a query on the entire data may still not get an accurate query result, and sampling further reduces result quality.
Data cleaning typically requires domain-specific software that can be costly and time-consuming to develop. Particularly, in order to obtain reliable cleaning results, many data cleaning techniques need humans to get involved. While crowdsourcing makes this increasingly feasible, it is still highly inefficient for large datasets. In this paper, we suggest an efficient methodology that has potential to improve answer quality.
The paper is organized as follows. In Section 2, we discuss all the related work to define dirty databases and the semantics of clean answers. In Section 3, Methodology how we perform the query rewriting. Finally, Section 4 concludes the paper by discussing the contribution and results.

Chapter 2

Related Work

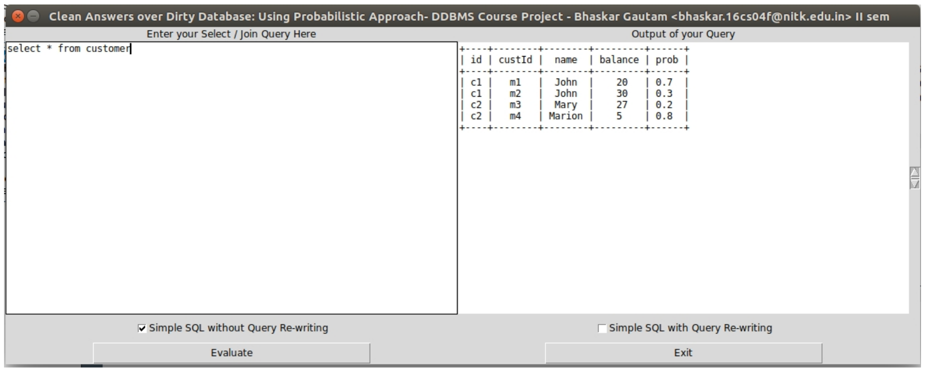
Schema Matching Matching individual attributes for similarity (whether value overlap or equivalence of semantic domains) is only one important task of many during schema matching. More comprehensive systems such as Bellman [5], Clio [8] or iMap also consider matches between attribute sets and simple transformations. Our work solves one specific task here, e.g. that of an 'overlap searcher' in iMap, with the capability of being able to find overlap between 'dirty' value sets. Due to our focus on dirty data, our methods can also match compound attributes (up to a degree), e.g. where 'location' matches 'City' plus 'State', and thus help reduce the search space. A difference (good or bad, depending on the application) to other approaches, e.g. [9], is that SoS-similarity aims at finding overlap in value-sets, rather than domain equivalence. After matching attributes have been identified by our approach, it is possible to design queries using approximate joins. For matching individual values (for oracle functions) or attribute names (for schema-based matchers), string similarity measures are needed. Here a number of different metrics have been proposed, including edit distance, affine gap distance, Smith-Waterman distance and q-gram distance.

Database Sampling Various approaches for sampling databases have been proposed. One important application is the estimation of join sizes for query optimizations. However, despite the fact that we are looking for approximate join candidates, we are not so much interested in estimating the join size of these queries, but are looking for similarity of their value sets. Join size is not such a good indicator for us, since a single pair of similar frequent values (or absence of such a pair) can have a huge impact.Thus any mismatch of values can easily create a very wrong similarity result. Furthermore, samples must either be very large to obtain accurate results, or otherwise require careful examination of both sets to ensure that the 'right' values are selected, which is inefficient for more than 2 attributes. Other sampling approaches aim to estimate aggregation results. As they focus on a single attribute only, they are not applicable to our problem. The issue of page-level sampling vs row-level sampling is addressed in, but is not directly relevant to our work. [10] deal with sampling methods based on query operators. For us, the relevant operator is (approximate) set intersection. However, set intersection is not handled in [11], while [12] concludes that independent random sampling “is so in efficient that it is rarely worth while”.

Chapter 3

Methodology

Can computers learn to read? We think so. Through this paper we attempts to create a database management system system that learns from the user query inorder to give the clean answers over the dirty database.Such amalgamation of data from different data sources discusses in below section. We assume that the dirty database that we use for answering have been classified into seperate clusters by some data integration tool whose mail goal is to form different clusters.


Clean Answers over Dirty Database: Using Probabilistic Approach- DDBMS Course Project - Bhaskar Gautam <bhaskar.16cs04f@nitk.edu.in> II sem
Enter your Select / Join Query Here
Output of your Query
Evaluate
Exit

```
select * from customer
```

```
+----+--------+--------+---------+------+
| id | custId |  name  | balance | prob |
+----+--------+--------+---------+------+
| c1 |   m1   |  John  |    20   | 0.7  |
| c1 |   m2   |  John  |    30   | 0.3  |
| c2 |   m3   |  Mary  |    27   | 0.2  |
| c2 |   m4   | Marion |    5    | 0.8  |
+----+--------+--------+---------+------+
```

[✓] Simple SQL without Query Re-writing
[ ] Simple SQL with Query Re-writing

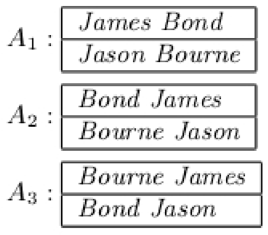
Figure 3.1: A Clean Answers over Dirty Databases Simulator

3.1 Sampling dirty data for matching attributes

In real world, when we are integrating data from different sources, data sets are dirty which arises problem to identify relationship between tables and attributes. There are number of algorithmic approaches for determining such relations, they are based on identification of matching attributes. These matchers are of two types, schema-based matchers and instance based matchers. Schema

based matchers rely on schema information only while instance based matchers rely on actual data sets, they are slower but more accurate than schema based ones. Schema based matchers are limited in power, they are based on domain equivalence where attributes with common values having same semantic domain e.g. name, company, location etc.
For data integration, it is more relevant to find attributes pair with matching domains but with this value overlap is also important, as tables carrying informations about same entities should be integrated differently via join on such attributes from tables carrying information about different entities from same domain via union.
To deal with large data values of distributed data we use sampling techniques. Sampling can be done locally and only sample set is required to transmit , when all sampling sets are collected, they can be analyzed to find matching attributes. Sampling can also speed up computation significantly if data is not distributed. To resolve the problem of dirty data, a sampling technique of classic sampling techniques is introduced which preserves synchronization property and making sample sizes smaller than random sampling sizes, new similarity measures are introduced which can be used to match string with small differences and can be computed easily from samples, by comparing sets of sets which are obtained by transforming each string in attribute's value set into a set of strings.

Example 1. Consider the following three (very small) value sets, corresponding to three different attributes: If we form their 3-gram sets as described above



A_1 :	*James Bond*
	Jason Bourne
A_2 :	*Bond James*
	Bourne Jason
A_3 :	*Bourne James*
	Bond Jason

(using only substrings of words), we get
A1 :{Jam,ame,mes,Bon,ond,Jas,aso,son,Bou,our,urn,rne}
A2 :{Bon,ond,Jam,ame,mes,Bou,our,urn,rne,Jas,aso,son}
A3 :{Bou,our,urn,rne,Jam,ame,mes,Bon,ond,Jas,aso,son}
Here A2 and A3 map to exactly the same sets of chunks, and thus appear to be identical and equally similar to A1 ,although A2 should really be considered closer to A1 and somewhat different from A3.
What we propose is a new measure for attribute similarity. For this we also map each string value to the set of its chunks, but don't combine all chunk sets into one. For the tables from Example 1 we then get
A1 :{{ Jam,ame,mes,Bon,ond }, { Jas,aso,son,Bou,our,urn,rne }}
A2 :{{ Bon,ond,Jam,ame,mes }, { Bou,our,urn,rne,Jas,aso,son }}
A3 :{{ Bou,our,urn,rne,Jam,ame,mes }, { Bon,ond,Jas,aso,son }}

For finding matching attributes compute similarity for all pairs of value sets, Jaccrd Index or resemblance is well known measure for similarity of sets the resemblance of two sets α, β is define in below equation

$$res(\alpha, \beta) = \frac{|\alpha \cap \beta|}{|\alpha \cup \beta|} \tag{3.1}$$

And in sample generating, sample should be synchronized.
Sampling should be synchronized, i.e., if a particular value x is sampled for A then it is also sampled for B (provided it lies in B), and vice versa:

$$x \in S(A) \quad and \quad x \in B \quad \Rightarrow \quad x \in S(B) \tag{3.2}$$

3.2 Intersection Resemblance Sum: IR-SUM

We are looking for a similarity measure which can be approximated using samples of small size. In particular, if we want to allow unbiased estimates using sample size one, our measure for sets (of sets) α, β must take the form

$$F(\alpha, \beta) = \sum_{a \in \alpha, b \in \beta} f(a, b) \tag{3.3}$$

for some function f(a,b). While there is no inherent requirement to allow samples of size one (which would be extremely unreliable), above equation still gives us an idea for how to construct our similarity measure in general.

Definition 1. (IR-Sum). Given two sets-of-sets α, β the Intersection-Resemblance-Sum of A, B is

$$IR\sum(\alpha, \beta) := \sum_{a \in \alpha, b \in \beta} |a \cap b|.res(a, b) = \sum_{a \in \alpha, b \in \beta} \frac{|a \cap b|^2}{|a \cup b|} \tag{3.4}$$

Definition 2 (SoS-resemblance). We define the Set-of-Set-resemblance, short SoS-resemblance, as

$$SoS - res(\alpha, \beta) := \frac{IR\sum(\alpha, \beta)}{IR\sum(\alpha, \alpha) + IR\sum(\beta, \beta) - IR\sum(\alpha, \beta)} \tag{3.5}$$

Chapter 4

Contribution and Results

We enhance the efficiency of probability calculation algorithm by not only considering the type of attributes used in schemas but also consider the tuple values. Due to which probability of getting clean answer is increase. Further we propose the new probability generation formation (4) whose performance is thoroughly evaluated by Levenshtein distance measure and with the Longest Common Substring score.

$$LevensteinDistanceScore(\alpha, \beta) = \frac{LevensteinDistance(\alpha, \beta)}{Length.(\alpha + \alpha)} \tag{4.1}$$

$$LongestCommonSubstringScore(\alpha, \beta) = \frac{Length.LCS(\alpha, \beta)}{Length.(\alpha + \beta)} \tag{4.2}$$

Shannon Diversity Index are used by the authors in the base paper of this project. They used this index to find the probability between attributes of two different schemas.

$$ShannonDiversityIndex(\alpha, \beta) = \frac{n * log(n) - \sum f_i * log(f_i)}{n} \tag{4.3}$$

We extend this approach by genrating probability based on the attributes values also and for that we describe the formula for the same

$$distance_d(\alpha, \beta) = \frac{max.Len(Tuple_{attrib}) - [len(\alpha) - len(\alpha \cap \beta)]}{max.Len(Tuple_{attrib})} \tag{4.4}$$

Theorem 1 *Let q be a rewritable query. Then, RewriteClean(q) computes the clean answers for q on every dirty database.*

The first step of the algorithm computes the cluster representatives and initializes the sum of distances S in each cluster, which will be used as a normalization factor to compute the probabilities. For each tuple t within a particular cluster with identifier c_i , Step 2 calculates the distance of this tuple to the representative, rep_i , of its cluster and adds this distance to S(c_i). Finally, in order

to compute the probability of a $tuple_t$, Step 3 turns the distance computed in the previous step for this tuple into a similarity.
To evaluate the perfomance of our mathematical relation, we compare our re-

```
Function Tuple Prob ;
Input  : A set of tuples T,
         - a clustering C = $c_1$, $c_2$, .... $c_k$ of T,
         where $c_i$ is the identifier of cluster $c_i$
         - a distance measure d
Output: For every tuple t in T, a probability prob(t)
while $i!=k$ do
    - Compute cluster representative $rep_i$ for $c_i$ by merging all the tuples
      that belong to it
end
while $tuple\ t\ \varepsilon\ T\ and\ t\ \varepsilon\ c_i$ do
    - Compute $d_t$ = d(t,$resp_i$), the distance of t to the representative of
      its cluster. - Add $d_t$ to S($c_i$)
end
while $tuple\ t\ \varepsilon\ T\ and\ t\ \varepsilon\ c_i$ do
    - Compute similarity $s_t = 1 - \frac{d_t}{S(c_i)}$
    - prob(t) = 1.0 if $|c_i| = 1$, or
    prob(t) = $\frac{s_t}{|c_i|-1}$ otherwise
end
```

Algorithm 1: Assigning Tuple Probabilities

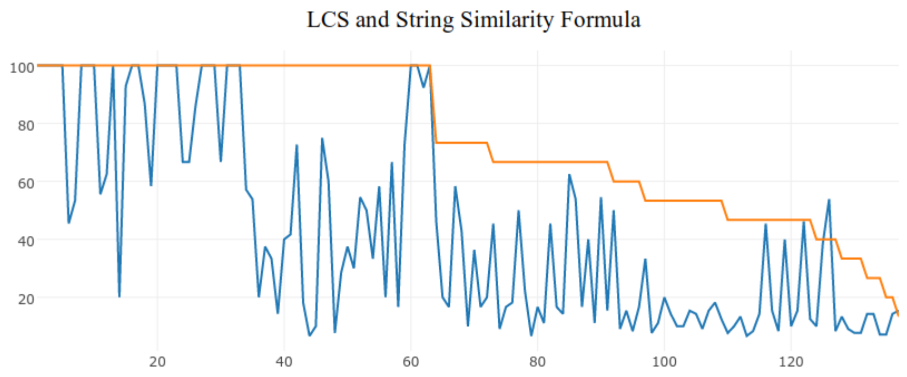


Figure 4.1: Relation between our String matching and Longest Common Substring.

lation with many distance relations apart from Shannon Diversity Index. First our evaluation model preceeds with comparing the distance with Levenstein

Distance between two attribute values. Finally we compare our relation with

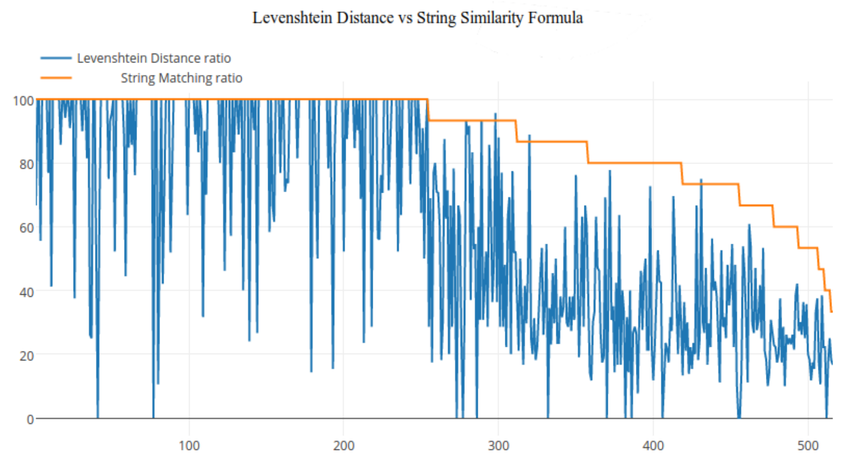


Figure 4.2: Relation between our String matching and Levenstein Distance ratio.

the Longest Common Subsequence score. Further we conclude that the performance of our relation is very high as it does not fluctuate the result when minute characters are missing.

Bibliography

[1] `Bhaskar Gautam, https://github.com/bhaskar24/Clean_Answers_over_Dirty_Database`

[2] R. Ananthakrishna, S. Chaudhuri, and V. Ganti, *Eliminating fuzzy duplicates in data warehouses* , In VLDB, pages 586–597, 2002.

[3] P. Andritsos, A. Fuxman, and R. J. Miller, *Clean answers over dirty databases(Technical report)*,UofT, Dept of CS, CSRG-513, 2005.`ftp://www.cs.toronto.edu/csri-technical-reports/513/tr513.pdf`.

[4] P. Andritsos, R. J. Miller, and P. Tsaparas, *Information-Theoretic Tools for Mining Database Structure from Large Data Sets*, In SIGMOD, pages 731–742, 2004.

[5] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik. LIMBO, *Scalable Clustering of Categorical Data*, In EDBT,pages 123–146, 2004.

[6] T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk, *Mining database structure; or, how to build a data quality browser*, in SIGMOD, 2002, pp. 240–251.

[7] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik. LIMBO, *Scalable Clustering of Categorical Data*, In EDBT,pages 123–146, 2004.

[8] R. J. Miller, L. M. Haas, and M. A. Hernández, *Schema mapping as query discovery*, in VLDB, 2000, pp. 77–88.

[9] R. Dhamankar, Y. Lee, A. Doan, A. Y. Halevy, and P. Domingos, *iMAP: Discovering complex mappings between database schemas*, in SIGMOD, 2004, pp. 383–394.

[10] B. T. Dai, N. Koudas, D. Srivastava, A. K. H. Tung, and S. Venkatasubramanian, *Validating multi-column schema matchings by type*, in ICDE, 2008, pp.120–129.

[11] W. W. Cohen, *Integration of heterogeneous databases without common domains using queries based on textual similarity*, in SIGMOD, 1998, pp. 201–212.

[12] L. Gravano, P. G. Ipeirotis, H. V. Jagadish,N. Koudas, S. Muthukrishnan, and D. Srivastava, *Approximate string joins in a database (almost) for free*, in VLDB, 2001, pp. 491–500.