Utility of Considering Multiple Alternative Rectifications in Data Cleaning

by

Preet Inder Singh Rihan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved September 2013 by the
Graduate Supervisory Committee:

Subbarao Kambhampati, Chair
Hasan Davulcu
Huan Liu

ARIZONA STATE UNIVERSITY

December 2013

ABSTRACT

Most data cleaning systems aim to go from a given deterministic dirty database to another deterministic but clean database. Such an enterprise pre-supposes that it is in fact possible for the cleaning process to uniquely recover the clean versions of each dirty data tuple. This is not possible in many cases, where the most a cleaning system can do is to generate a (hopefully small) set of clean candidates for each dirty tuple. When the cleaning system is required to output a deterministic database, it is forced to pick one clean candidate (say the "most likely" candidate) per tuple. Such an approach can lead to loss of information. For example, consider a situation where there are three equally likely clean candidates of a dirty tuple.

An appealing alternative that avoids such an information loss is to abandon the requirement that the output database be deterministic. In other words, even though the input (dirty) database is deterministic, I allow the reconstructed database to be probabilistic. Although such an approach does avoid the information loss, it also brings forth several challenges. For example, how many alternatives should be kept per tuple in the reconstructed database? Maintaining too many alternatives increases the size of the reconstructed database, and hence the query processing time. Second, while processing queries on the probabilistic database may well increase recall, how would they affect the precision of the query processing? In this thesis, I investigate these questions. My investigation is done in the context of a data cleaning system called BayesWipe that has the capability of producing multiple clean candidates per each dirty tuple, along with the probability that they are the correct cleaned version. I represent these alternatives as tuples in a tuple disjoint probabilistic database, and use the Mystiq system to process queries on it. This probabilistic reconstruction (called BayesWipe-PDB) is compared to a deterministic reconstruction (called BayesWipe-DET)–where the most likely clean candidate for each tuple is chosen, and the rest of the alternatives discarded.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Subbarao Kambhampati, for giving me an opportunity to work under his guidance in his elite group, Yochan. It is the result of his guidence that I have been able to successfully complete my Masters.

I would like to express my gratitude to Dr. Hasan Davulcu and Dr. Huan liu for their guidance and valuable feedbacks. Their exceptional co-operation and guidance made the whole dissertation process smoother.

Importantly, I would like to thank Sushovan De for mentoring me and providing invaluable inputs and immense help during the course of my masters research. I am also very thankful to all my colleagues in Yochan group – Kartik Talamadupula, Srijith Ravikumar, Tuan Anh Nguyen, Yuheng hu, Mi Jun for taking time out of their own work and helping me by providing valuable feedbacks that helped me in improving the quality of my research.

I am grateful to Michael Sullivan for his help and support during my studies. I learned a lot from him. I would also like to thank my friends Kanchan, Tejas, Vishnu, Dhruv, Arun, Shruti, Kuhu, Vaijan and Richa for their encouragement and support.

My special thanks to my father, my mother and my sister for their continuous support, encouragement and love without which I could never have reached this far.

TABLE OF CONTENTS

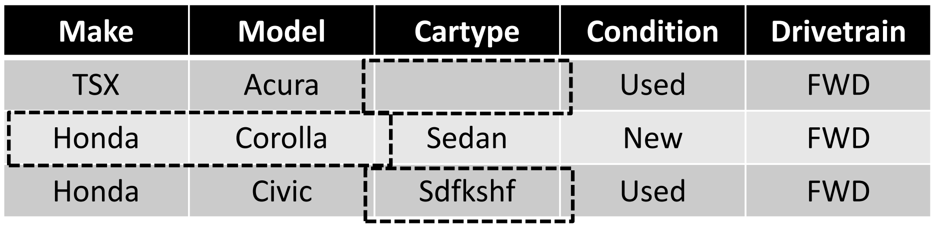
LIST OF FIGURES

Chapter 1

Introduction

1.1 Data Cleaning: Problem

The importance of data has grown tremendously over the last decade. Data plays a pivotal role in many processes. This includes crucial decision making and analysis processes. For example, with the introduction of eScience [9], analysis of immense data has become a primary method of many new scientific discoveries. The volume, variety and velocity of data that is being acquired today, have reached an unprecedented level. Although advancements in computational technology have provided efficient ways to store the data but the success of all data centric processes depends primarily upon the validity of the data available for operations. For instance, when using data in decision making processes, quality of data is vital in order to avoid wrong conclusions. There could be many reasons behind the presence of a noise in data e.g. imperfect sensing devices and/or information extractor, heterogeneity in data coming from different sources, misspelling during data entry etc.[5] All these reasons may give rise to many data quality problems such as missing values, inconsistent values, schema inconsistencies, data duplication etc. Data that contains such quality problems is referred to as dirty data.



Make	Model	Cartype	Condition	Drivetrain
TSX	Acura		Used	FWD
Honda	Corolla	Sedan	New	FWD
Honda	Civic	Sdfkshf	Used	FWD

Figure 1.1: Dirty Data.

Example: The sample table shown above is a snapshot of the database containing inconsistencies and inaccuracies in the values.First tuple has an error due to incompleteness. Second and third tuples have error due to inconsistencies e.g. corolla is not honda car, and wrong entry respectively.

Dirty data needs to be cleaned before it can be used in crucial processes. Today, the data cleaning problem is seen as a collection of separate problems (Duplicate removal, outliers detection, data inconsistencies etc.). Every problem in this set is a process which deals with detecting and repairing inaccuracies and inconsistencies present in dirty data [5].

1.2 Current State of Data Cleaning

Currently, there are several commercial tools available for handling errors and anomalies in data e.g. Informatica, Oracle Warehouse Builder etc. All these tools are referred to as ETL (Extraction, Transformation and Loading) tools [4, 21]. The effectiveness of these tools depends upon the cleaning specifications set by domain experts. Apart from that, a variety of data cleaning approaches have been suggested over the years (e.g., outliers detection [19], noise removal [24], entity resolution and imputation [13], recent effort on examining integrity constraints, e.g. functional/inclusion dependencies (FD/IND) and their extensions (CFD/CIND) [7, 12, 3], and BayesWipe [15]).

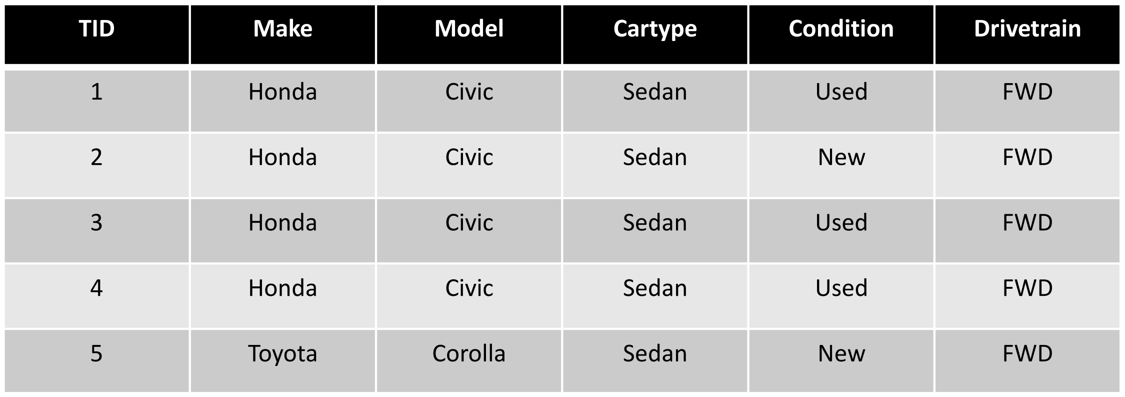
Each of the above data cleaning approaches is applicable to their own limited area of application, and comes with their own advantages and disadvantages. A common theme across all these approaches is the fact that despite evaluating multiple alternative rectifications of a dirty data tuple, only a single cleaned version is produced as an outcome. However, deciding a unique clean version of dirty data requires making a choice among multiple alternative rectifications of a dirty data instance. All these approaches tend to ignore uncertainties present in the cleaning process and resolve these uncertainties deterministically by using some fixed rules or domain expert inputs [4]. These kinds of data cleaning approaches generate deterministic clean outcomes. In current data cleaning approaches, only one clean version of the dirty data is produced. Since current approaches rely highly on the fixed rules/domain expert's knowledge, depending upon their quality, these approaches may introduce errors in the cleaned output, which results further irrecoverable loss. Furthermore, the variety and volume of data make it more difficult for generating effective set of fixed rules which are very critical to current data cleaning approaches. In many scenarios, it is almost impossible to recover from such corruptions in data when deterministic cleaned outcome of data cleaning approach is decoupled from original dirty data [4].

1.3 Proposed Approach For Storing Outcomes of Data Cleaning

By ignoring other alternative rectifications of a data tuple in order to produce a deterministic clean data tuple, current data cleaning approaches may end up rejecting valuable information that they have generated while data cleaning. An alternative approach to avoid such loss is to consider all alternative rectifications generated during data cleaning approach. This approach will produce stochastic outcomes as opposed to deterministic outcomes of current data cleaning approach. All uncertain rectifications of a data tuple will be stored in a probabilistic database. Belief, in using a probabilistic database to handle incompleteness and errors in data after data cleaning, is strengthened due to a suggestion made by CRA [2]. Also, it has been shown that maintaining the uncertainties in the specific problem of data cleaning i.e. data de-duplication does not involve too much overhead in space and time [6].

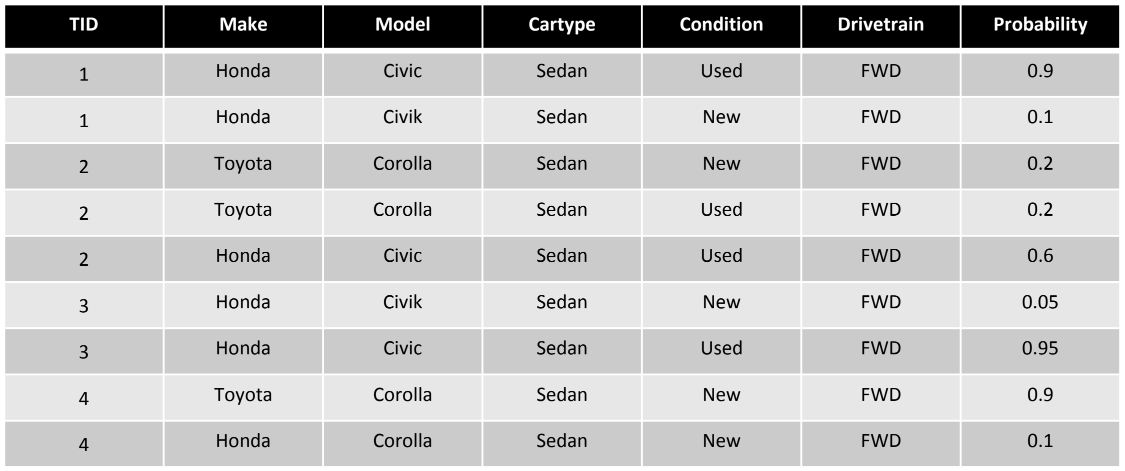
Although keeping multiple alternative rectifications of a data tuple seems to have some improvement e.g. better recall and prevention of corruption, over deterministically picking one rectification in data cleaning process, it is not intuitive if this proposed approach would make any significant improvements when a larger probabilistic database is used in order to retain all alternative rectifications of a data tuple over a deterministic database that is of the same size as that of original dirty data. On the positive side, it looks as if considering all rectifications will help in avoiding the loss of information caused by deterministically selecting one ultimate clean version. But on the other hand, probabilistic database may lose to deterministic database on the question of scalability i.e. higher query processing times and larger physical space requirements. In addition to this, handling large number of irrelevant rectifications present in probabilistic database poses another severe challenge.

In this work, we investigate the trade-offs of considering multiple alternative rectifications of a dirty data instance against having a deterministically selected unique clean version. For investigation, BayesWipe – a data cleaning system for structured data– is used. BayesWipe generates two different sets of outcomes after cleaning a structured dirty data. BayesWipe-PDB, a set of stochastic outcomes from BayesWipe is compared against the BayesWipe-DET, deterministic outcome of the same system. BayesWipe-PDB contains all alternative rectifications of a dirty data tuple where as BayesWipe-DET considers one deterministically picked rectifications of dirty data tuple. I investigate trade-offs of BayesWipe-PDB and BayesWipe-DET by comparing them on scalability and usefulness of their query result set.



TID	Make	Model	Cartype	Condition	Drivetrain
1	Honda	Civic	Sedan	Used	FWD
2	Honda	Civic	Sedan	New	FWD
3	Honda	Civic	Sedan	Used	FWD
4	Honda	Civic	Sedan	Used	FWD
5	Toyota	Corolla	Sedan	New	FWD

Figure 1.2: BayesWipe-DET, BayesWipe's Deterministic outcomes



TID	Make	Model	Cartype	Condition	Drivetrain	Probability
1	Honda	Civic	Sedan	Used	FWD	0.9
1	Honda	Civik	Sedan	New	FWD	0.1
2	Toyota	Corolla	Sedan	New	FWD	0.2
2	Toyota	Corolla	Sedan	Used	FWD	0.2
2	Honda	Civic	Sedan	Used	FWD	0.6
3	Honda	Civik	Sedan	New	FWD	0.05
3	Honda	Civic	Sedan	Used	FWD	0.95
4	Toyota	Corolla	Sedan	New	FWD	0.9
4	Honda	Corolla	Sedan	New	FWD	0.1

Figure 1.3: BayesWipe-PDB, BayesWipe's Stochastic Outcomes

Example: The example above shows a snapshot of both BayesWipe-DET and BayesWipe-PDB. In BayeWipe-DET, one rectification is generated for every dirty data tuple, whereas in BayesWipe-PDB, a dirty data tuple can have more than one stochastic rectifications with each having a probability value associated with it. Probability value shows systems confidence for claiming a particular rectification to be true tuple in real world.

1.4 Thesis Organization

My thesis is organized into six chapters. Chapter 2 provides a summary of related work which considers multiple alternatives after data cleaning and querying over imprecise data. As my investigation is based upon a probabilistic data cleaning system, BayesWipe and a probabilistic databases, I cover a brief overview of BayesWipe and MystiQ, a probabilistic database management system, in Chapter 3. Chapter 4 discusses the detailed approach followed by me for evaluation of BayesWipe-PDB against BayesWipe-DET. BayesWipe-PDB is investigated against BayeseWipe-DET over multiple experiments. A detailed description of setup and findings of each experiment are explained in Chapter 5. I conclude my work in Chapter 6.

Chapter 2

Related Work

Data Cleaning problems have received significant attentions from traditional database community and many approaches have been proposed to address different data cleaning problems e.g. missing values, inconsistencies, duplicate detections etc.

Most data cleaning approaches[7],[15],[17] etc. consider only one rectification of the dirty data instance. [7] uses a minimal cover algorithm to reduce the redundant conditional functional dependencies, where as [15] considers only that rectification that has maximum probability score. [17] considers outiler based on maximum distance and ignores the rest. One exception is the approach mentioned in ProbClean [6] . This paper describes a system that takes into accounts multiple alternative rectifications from the data cleaning process. But this paper emphasizes on another data cleaning problem which focuses on removing duplicates from the database. It maintains multiple repairs of dirty data in order to capture uncertainties in cleaning process. ProbClean, focuses on scalability of the approach but it does not evaluate system on accuracy against a data cleaning procedure which ignores the uncertainties involved. It compares probabilistic duplicate detection approach with deterministic duplicate detection approach for clustering time, output size, query running time and queries overhead against the data size and percentage of duplicate (Noise in Data).

MystiQ[8] shows a way to execute a SQL query in order to obtain the results from probabilistic database. In this work, to check the quality of information stored in probabilistic database multiple random queries are run over the probabilistic data. This probabilistic query answers needs to be evaluated. There are quite a few number of notable works [11], [10] and [18] that query the imprecise data but I am not aware of any technique that compares the imprecise data results with the precise data results using common metrics.

Chapter 3

Background

In my thesis, I investigate usefulness of Probabilistic outcomes of a Data Cleaning approach. My investigation is based on BayesWipe–Probabilisitic data cleaning system– developed by our research group. I choose this system since it has a tendency to generate multiple clean versions of a single dirty tuple with a probability value associated with each clean version. Also as it developed by our group, I have the full controll over the system. Current implementation of BayesWipe resolves uncertainties among multiple versions by following MLE (Maximum likelihood Estimation Approach). Here is the brief overview of BayesWipe [15].

3.1 BayesWipe Overview

BayesWipe is an end-to-end probabilistic framework for data cleaning process. It views data cleaning problem as a statistical inference problem over structured data. It learns two models from the noisy data – data source model and data error model. Data source model is a learned model of clean data generation process where as data error model learns the corruption process of how the noise is introduced. By treating clean values as a latent random variable, BayesWipe leverages these two learned model and infers its value through Bayesian estimation. [15].

Let $D = T_1.....T_n$ be the input data set that may contain tuple with corrupted attributes. BayesWipe first finds a set of correction candidate C for given T_i. $C = T^*|AttrDiff(T^*, T) < \Delta$ In this correction candidate set, each T* differs from T in at most a threshold Δ number of attributes. After that it will calculate $P(T^*|T)$ for each tuple in candidate set and return the one which has highest $P(T^*|T)$ as the clean version of T.

$$T^* = \arg\max P(T|T^*) \text{ for every } T^* \in C$$

To calculate $P(T^*|T)$ for every T*, it uses Bayes rule as

$$P(T^*|T) = P(T|T^*).P(T^*)$$

$P(T^*)$ and $P(T^*|T)$ are calculated from data source models and data error models respectively.

Data Source Model

This is a generative model, which gives a probability distribution over clean version of tuples i.e. P [T*], to do so it needs to generate all m attributes of the schema of Dataset the D. Generally in structured data, there could be number of dependencies among attributes. This data model captures all those dependencies in a Bayes Network. Bayes Network is constructed in two steps. In first step BayesWipe obtains the topology from a tool named Banjo [14]. Then after learning the topology BayesWipe estimates the conditional probability tables that parameterize each node in learned topology. This is done with a tool called Infer.NET [20, 15].

Data Error Model

BayesWipe estimates error model $P(T|T*)$ from noisy data. Given a set of candidate clean tuples it estimates how similar is T to T*. BayesWipe makes some assumptions regarding the error in the data such as errors are combination of spelling, incompletion and substitution errors. BayesWipe defines its own similarity function as it cannot use regular string similarity measures on tuples which is a composition of independent string. BayesWipe$'$s similarity function includes two similarity measures: edit distance similarity f_{ed} and distributional similarity f_{ds}, and it generates a unified error models. [15].

Apart from this, BayesWipe [15] also maintains an error statistic that is used at query time. It uses error statistics to predict the likelihood that a particular attribute value in that tuple is misspelled, substituted or missing.

3.2 MystiQ

Probabilistic databases provide a great framework to handle uncertainty in data. The uncertainty is expressed in terms of probabilities – a tuple is present in the database with some probability, or the value of attribute is given by some probability distribution. I use a probabilistic database MystiQ [8] to store the outcome of the BayesWipe data cleaning process. It will store multiple possible clean tuples for every single tuple in the database. Every clean version of the corrupted tuple will have some probability associated with it that shows the degree of confidence in that particular tuple to be the clean version of the corrupted tuple. MystiQ [8], a Probabilistic database management system, is a research prototype developed at University of Washington. Like any other Probabilistic databases, it extends traditional database technology to handle uncertainties. Uncertainties are expressed in terms of probabilities [23]. It supports query as complex as those supported by advance query processors of traditional databases. It allows this while considering the data to be uncertain. When SQL query is executed on it, MystiQ will return set of results where each answer is annotated with a probability.

Chapter 4

Approach for Investigating the Utility of Multiple Alternative Rectifications in Data Cleaning

I conduct a detailed study in order to evaluate utility of retaining multiple alternative rectifications of dirty data tuple in a probabilistic database. I compare this probabilistic database against a deterministic database that stores only deterministic outcomes. First, I create two different outcomes of the BayesWipe data cleaning system, BayesWipe-DET and BayesWipe-PDB. BayesWipe-DET considers only most likely rectification for a dirty data tuple, whereas BayesWipe-PDB considers multiple alternative rectifications for a dirty data tuple. Then an investigation strategy is developed in order to evaluate the advantages and disadvantages of BayesWipe-PDB and BayesWipe-DET.

4.1 BayesWipe-DET

BayesWipe-DET is a deterministic database created from the outcomes of the BayesWipe data cleaning process. It retains only one deterministically picked rectification per dirty data tuple. For every tuple T in dirty database, BayesWipe generates a set of candidate replacement tuples i.e a set of T*. Every candidate replacement tuple will have a probability associated with it which is the likelihood of that tuple to be the actual clean version in the real world. To generate BayesWipe-DET, these stochastic results of BayesWipe are converted to deterministic results by picking only one candidate replacement tuple which has highest likelihood of becoming the clean version of the dirty tuple in real world. All other proposed rectifications are rejected. That candidate replacement tuple is stored into a relational database that is called BayesWipe-DET. Figure 4.1 shows the BayesWipe-DET generation process.

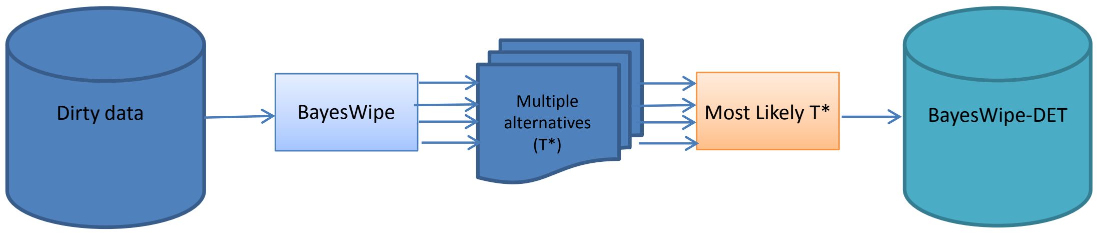


Figure 4.1: BayesWipe-DET Generation

4.2 BayesWipe-PDB

BayesWipe-PDB is the outcome of BayesWipe data cleaning process which retains all alternative rectifications for a dirty data tuple generated during data cleaning. As mentioned above, BayesWipe performs the data cleaning tuple by tuple. It takes a dirty data tuple and transforms that dirty tuple into a set of stochastic candidate replacement tuples. Every dirty tuple can be one of the candidate replacement tuple and the probability of that is given by a probabilistic distribution $P(T^*|T)$ generated by BayesWipe data cleaning system. Such stochastic outcomes of BayesWipe data cleaning system are perfectly in accordance with the definition of probabilistic database whose meaning is quite simple. Probabilistic database means that an instance of the database can be in several states and each state will have a probability value associated with it [23]. To consider multiple alternative rectifications in data cleaning, a probabilistic database is created and is called BayesWipe-PDB. BayesWipe-PDB is stored in relational database and MystiQ, [8] a probabilistic query processing engine is used to interact with probabilistic database.

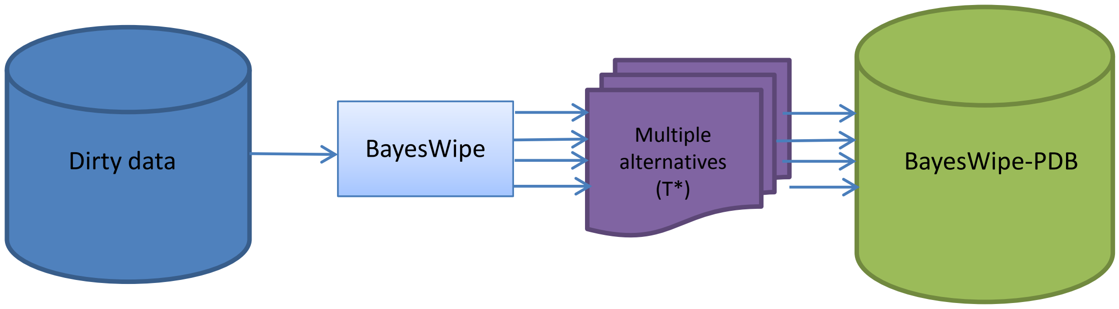
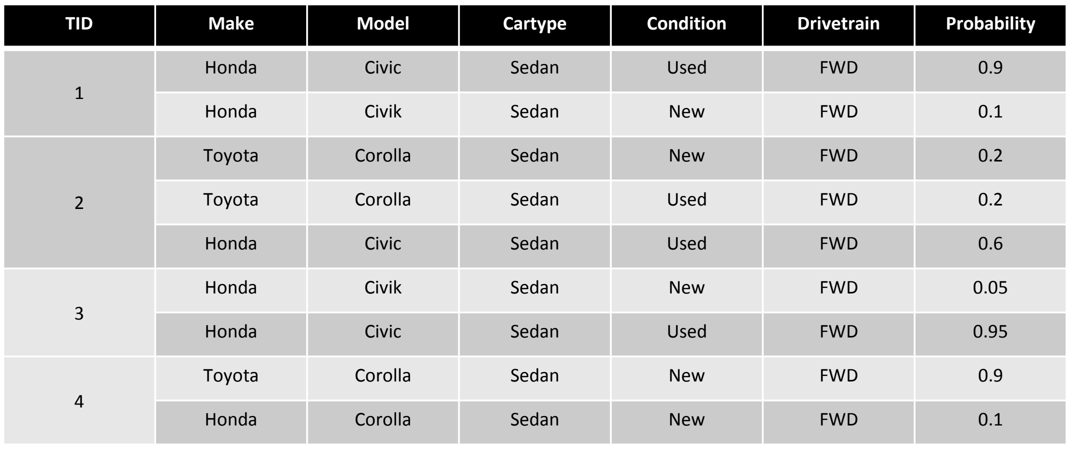


Figure 4.2: BayesWipe-PDB Generation

The probabilistic database generated from BayesWipe data cleaning system has uncertainty present at tuple level [23]. In tuple level uncertainty, a tuple is a random variable and tuple′s belongingness in the database is given by a probability value. A probabilistic database is simply a probability distribution over set of possible worlds of the full database [23]. A state of the probabilistic database where each data tuple is present in one of its many forms, is known as possible world of a probabilistic database. In this case, number of possible worlds for a probabilistic database created from BayesWipe is exponential in number of tuples in dirty database.

In practice, there are two ways to represent the probabilistic database in relational database i.e. tuple independent probabilistic database and block disjoint independent probabilistic database. In tuple independent probabilistic database, tuples are independent probabilistic events [23]. But, the best possible representation of a probabilistic database generated from outcomes of BayesWipe data cleaning is block disjoint independent(BID), that is also known as tuple disjoint independent. In BID, possible tuples are partitioned into blocks and all tuples in a block represent disjoint probabilistic events and tuples in different blocks are independent events [23]. BayesWipe produces multiple clean versions T* for a tuple T and all these T* of T are disjoint probabilistic events. These events are independent of all the T* of other tuples and so a set of T* represent a block of BID probabilistic database.



TID	Make	Model	Cartype	Condition	Drivetrain	Probability
1	Honda	Civic	Sedan	Used	FWD	0.9
	Honda	Civik	Sedan	New	FWD	0.1
2	Toyota	Corolla	Sedan	New	FWD	0.2
	Toyota	Corolla	Sedan	Used	FWD	0.2
	Honda	Civic	Sedan	Used	FWD	0.6
3	Honda	Civik	Sedan	New	FWD	0.05
	Honda	Civic	Sedan	Used	FWD	0.95
4	Toyota	Corolla	Sedan	New	FWD	0.9
	Honda	Corolla	Sedan	New	FWD	0.1

Figure 4.3: Block disjoint independent probabilistic database.

Example: The above example shows a snapshot of a Block Independent disjoint type of probabilistic database produced after retaining all alternative rectifications of dirty data tuple. Each tuple ID represents a unique tuple which now can be present in multiple forms. Tuples inside the blocks are mutually exclusive whereas outside the blocks are independent of each other.

4.3 Investigation Strategy

To find the trade-offs of considering multiple rectifications against keeping only one deterministically selected rectification, I follow an investigation strategy which evaluates BayesWipe-PDB against BayesWipe-DET on two different criteria. First, I evaluate BayesWipe-DET against BayesWipe-

PDB on the question of scalability. In this evaluation, I inspect BayesWipe-PDB and BayesWipe-DET for query processing time and physical space requirement. Second, I examine whether retaining all alternative rectifications has any significant impact on the usefulness of the query results when BayesWipe-PDB is used to provide query results instead of BayesWipe-DET.

4.4 Approach to evaluate BayesWipe-PDB and BayesWipe-DET on Scalability

In this evaluation, I compare BayesWipe-PDB and BayesWip-DET on the question of scalability. Since BayesWipe-PDB stores all alternative rectifications of a dirty data tuple, size of probabilistic database can be significant larger than deterministic database which stores only the most likely rectification. This will affect the performance of BayesWipe-PDB system in terms of physical space and query processing time. I examine if BayesWipe is still scalable to when all alternative rectifications are retained after cleaning a bigger dirty database. I also compare the query processing time over both BayesWipe-PDB and BayesWipe-DET.

4.5 Approach to evaluate BayesWipe-PDB and BayesWipe-DET on usefulness of query results

The motive of a Data cleaning process is to produce high quality data so that whenever cleaned data is used to get information for data analysis tasks, the extracted information is more reliable than the information from dirty data. In this evaluation, query results from BayesWipe-PDB are compared against query results obtained after running the same query on BayesWipe-DET. This helps in evaluating the fact that whether there is any benefit of keeping multiple alternative rectifications in a probabilistic database in terms of getting more useful information or not. An SQL select query system is used to extract results from both BayesWipe-PDB and BayesWipe-DET. I evaluate the precision and recall of query results from BayesWipe-PDB and BayesWipe-DET against the ground truth.

Query results from BayesWipe-DET

A SQL select query is used to get query result from deterministic database, BayesWipe-DET. SQL-Server is used as query engine and it runs the query over BayesWipe-DET like any other traditional relational database. Query results obtained are deterministic in nature – there is no uncertainty present at tuple level and every tuple constitutes the result set as a whole. Figure 4.4 shows the deterministic results obtained from BayesWipe-DET for a selection query $\sigma_{make='Honda'}$

TID	Make	Model	Cartype	Condition	Drivetrain
1	Honda	Civic	Sedan	Used	FWD
2	Honda	Civic	Sedan	New	FWD
3	Honda	Civic	Sedan	Used	FWD
4	Honda	Civic	Sedan	Used	FWD

Figure 4.4: Deterministic query results from BayesWipe-DET

Query results from BayesWipe-PDB

MystiQ [8] is used as a query processing engine to obtain results for an SQL select query from BayesWipe-PDB. This query result set is essentially of different kind from the query results generated by BayesWipe-DET. BayesWipe-PDB produces a result set that is probabilistic in nature. Since BayesWipe-PDB contains multiple alternative rectifications for a single tuple, it is very natural that more than one alternative rectifications are returned as a response to the query. Therefore, a query result set may have multiple instances of the same tuple. Each instance of the tuple has a probability associated with it which shows the belongingness of that particular instance in result set. Below figure 4.5 shows the probabilistic results obtained from BayesWipe-PDB for a selection query $\sigma_{make='Honda'}$

Query result – evaluation challenges

Precision and recall metrics are used to compare query results from BayesWipe-PDB and BayesWip-DET. Precision and Recall of query results is calculated against ground truth using standard precision recall formulas.

$$Precision = \frac{Truepositives}{TruePositives+Falsepositives}$$

TID	Make	Model	Cartype	Condition	Drivetrain	Probability
1	Honda	Civic	Sedan	Used	FWD	0.9
	Honda	Civik	Sedan	New	FWD	0.1
2	Honda	Civic	Sedan	Used	FWD	0.6
3	Honda	Civik	Sedan	New	FWD	0.05
	Honda	Civic	Sedan	Used	FWD	0.95
4	Honda	Corolla	Sedan	New	FWD	0.1

Figure 4.5: Probabilistic query results from BayesWipe-PDB

$$Recall = \frac{TruePositives}{TruePositives + Falsenegatives}$$

Query results evaluation is not straightforward. There are two major evaluation challenges associated with precision/recall evaluation.

1. *Defining membership of a result tuple in ground truth:* - When evaluating precision and recall of query result from cleaned data, an important concern is identifying membership of the resulting tuple in the ground truth. It is tricky to define which resulting tuple will correspond to a true positive and which will not. For an instance, consider a tuple T is returned as a resulting tuple from cleaned data. This tuple is returned since it's attribute values match query condition. This tuple has query attribute values same as that of the tuple from ground truth, but values of all remaining attributes may or may not be cleaned correctly. So, should this tuple T be considered as true positive or false positive. This is the question of relevance of tuple T that needs to be addressed beforehand.

   To resolve this issue, membership of a particular resulting tuple is defined in terms of tuple id only. Although this approach may look like a straightforward way to define membership, but this approach is intuitively correct. Consider running a query on a used-car database. When a user runs a query, she gets a set of results that the system has to determines relevant to her. As explained above, all resulting tuples may not have all attributes of the car cleaned correctly. The ultimate test of relevance of that car is when she actually evaluates the car in question by looking at the object in real life (and checking if it matches her criteria). Thus, the question of relevance is decided not by the attribute values in the database, but by the real-life entity that it is referring to. This reference to real-life entity is provided by the primary key (a tuple-id or a Vehicle Identification Number). That is why tuple ids are considered while calculating the precision and recall of query results. Tuples whose tuple ids appear in result set from ground truth and results from BayesWipe-DET or probabilistic results from BayesWiple-PDB are considered as true positives.

2. *Precision/Recall evaluation on probabilistic query results:* - Evaluation on probabilistic query results from BayesWipe is not straightforward as that of the deterministic results from BayesWipe-DET. Once the memebership of query results in ground truth is defined, precision/recall for deterministic results can be calculated using standard formulas of these metrics. It is not obvious how to extend precision and recall metrics to probabilistic databases in an efficient manner, while respecting possible world semantics [16]. When a query runs on probabilistic database, it generates a set of probabilistic tuples. Each probabilistic tuple is a set of all its alternative rectifications that match query condition and these rectifications will have a probability value associated with them.

   Ideally, in order to find precision/recall of a probabilistic query result set, one has to find precision/recall of query results from every possible world of probabilistic database using standard precision/recall formulas. Then the overall precision/recall of probabilistic result set can be calculated as the aggregated sum of product of precision/recall and probability of every possible world. There is no efficient algorithm defined for calculating precision and recall over exponential numbers of possible worlds. Therefore, precision and recall of probabilistic results are calculated using approximate methods. There are two possible methods to compute approximate precision and recall.

   In first method, approximate precision and recall is calculated by taking into accounts the fractional belongingness of the probabilistic tuples in query results from BayesWipe-PDB. Fractional belongingness is given by the aggregated probability of probabilistic tuple in query results from BayesWipe. Aggregated probability of a probabilistic tuple is the sum of probabilities of all alternative rectification of that probabilistic tuple present in result set. Then it uses following formulas to compute approximate precision and recall.

$$ApproximatePrecision = \frac{\Sigma_{t \in TP} P(t)}{\Sigma_{t \in TP} P(t) + \Sigma_{t \in FP} P(t)}$$

$$ApproximateRecall = \frac{\Sigma_{t \in TP} P(t)}{\Sigma_{t \in TP} P(t) + \Sigma_{t \in FN} P(t)}$$

In the second method, approximate precision and recall is calculated using the traditional way[1]. In this method, a threshold value is set and aggregated probability values of probabilistic tuples of query results are compared against it. All the tuples which have aggregated probability values less than threshold are removed from the result set. In order to perform the precision and recall calculations, probabilistic result set of a query is converted to deterministic result set. The conversion is done by collapsing all alternatives of a tuple into one result tuple represented uniquely by a tuple id. It is important to note that this determinization process is distinct from the determinization done at data cleaning process i.e. the determinized probabilistic result set is different from deterministic result set obtained from BayesWipe-DET. In fact, the probabilistic results followed by determinization is superior to the deterministic results because it can retrieve alternatives that may not individually be the most probable alternative, but collectively match the user's query to create a tuple. This determinization is followed by a thresholding over the total probability of the collapsed tuple. After thresholding, the remaining tuples in deterministic results are considered deterministic and standard precision/recall formulas are used to calculate approximate precision and recall of probabilistic query results.

I could not find a theoretical justification for using standard precision recall formula to consider only partial belongingness of a tuple so I choose second method to calculate the approximate precision and recall of the probabilistic query results which follows the standard precision/recall formula.

Thresholding–controlling accuracy of Probabilistic query results

A query over probabilistic database generates probabilistic result set which has probabilistic tuple of various probability value. Deterministic result set generated from probabilistic result set stores total probability value of probabilistic tuples in result set. Total probability value of a probabilistic tuple represents the confidence of the system in claiming that particular tuple is a part of ground truth result set when query is run on true data. Since all alternative rectifications of a dirty data tuple are stored into probabilistic database, it is more likely that result set will have many probabilistic tuples which have very low value of aggregated probabilities. In other words, system does not have good confidence in claiming those tuples to be a part of ground truth result set. A probabilistic tuple with low probability is more likely to be a false positive in real world. Since deterministic metrics like precision and recall are used to evaluate result set of query over BayesWipe-PDB, more false positive results reduce usefulness of the result set. So it is important to remove them from the results set. All alternatives of probabilistic tuples which have very low value of aggregated probability, are removed from probabilistic results set in order to increase the usefulness of result set. By manipulating threshold value, precision and recall of the result set can be controlled. If threshold value is kept high, it will remove less likely tuples from the result set but this will decrease recall. A low threshold value will consider more numbers of tuples even though there are many tuples in which system does not have much confidence.

4.6 Potential issues of considering multiple alternative rectifications

When BayesWipe cleans dirty data, for every tuple T, it generates multiple candidate replacement tuples with each having a probability value associated with it. Probability value shows how likely a particular tuple is the ultimate clean version of the tuple in real world. But, when the data size increases, it is observed that BayesWipe also generates a bigger set of candidate replacement tuples for a dirty tuple T. This set includes many candidate replacement tuples which have vanishingly small probabilities and are less likely to become the true tuple in real world. If all candidate replacement tuples are stored after the data cleaning process, size of probabilistic database can be very large. Large sized probabilistic database would cause issues related to scalability of this approach. These issues include greater query processing time and higher space requirement. To handle such potential issues, I present an optimization technique. Next section discusses it in details.

4.7 Optimization Techniques

In order to avoid potential issues associated with the approach that keeps multiple rectifications of a dirty data tuple, an optimization technique is presented. This technique is called Pre-Pruning.

All candidate replacement tuples for a dirty data tuple are retained in probabilistic database. Pre-Pruning follows an additional filtering step. It generates a subset of all alternative rectifications and keeps only those rectifications which pass pre-pruning test.

Pre-Pruning

Since size of BayesWipe-PDB is larger than the size of BayesWipe-DET, it is quite obvious that BayesWipe-PDB will have greater query processing time and higher physical space requirements. For large dirty dataset, it is observed that size of BayesWipe become very large and BayesWipe-PDB performs very poorly on scalability as compare to BayesWipe-DET.

Pre-Pruning is an optimization technique that is presented to address potential scalability issues. In Pre-Pruning, all alternative rectifications generated by BayesWipe data cleaning system are not considered equally important. In order to avoid a very large numbers of alternatives, which have diminishing probability values stored in the probabilistic database, a pruning algorithm is applied to the set of candidate replacement tuples (T^*). This pruning algorithm eliminates one or more T^*s and generates a subset to be preserved in probabilistic database, both reducing the database size as well as improving query time.

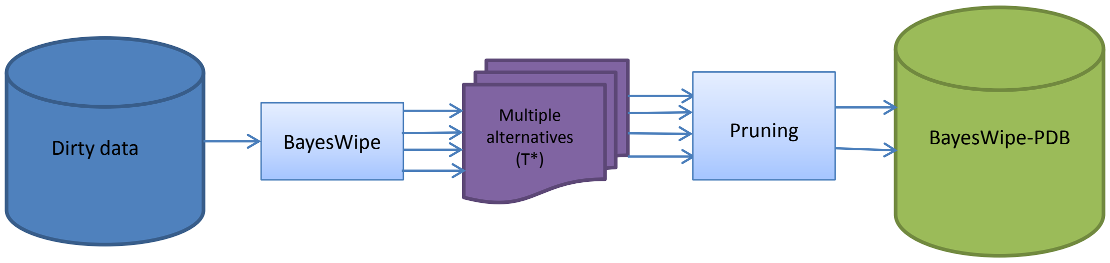


Figure 4.6: BayesWipe-PDB generation with Pruning

A naive method of performing pruning would be to place a single threshold on the probability, $P(T^*|T)$ and remove all alternative rectifications whose probabilities are less than threshold. However, it is observed from BayesWipe outcomes that keeping single threshold on $P(T^*|T)$ is not adequate, as there are many tuples that are legitimate tuples, but have a low score since they are infrequent in the database, or the data cleaning algorithm did not have enough error statistics to discover an error pattern. On the other hand, setting a fixed threshold low enough to allow all these clean rectifications into the probabilistic database almost removes any size advantage obtained from pruning. To perform efficient pruning, I come up with a novel algorithm to prune the set of T^*s to store in the Probabilistic database. The initial decision is based on $P(T^*|T)$.

Algorithm 1 Algorithm to prune set of candidate replacement tuples generated by BayesWipe

```
T*[]= set of all candidate clean version of a given tuple T
for $i = 0$ to $T^*.length$ do
    if ($P(T^*[i]|T) >= \beta$) then
        Keep T* as candidate clean version of T
    else if ($P(T^*[i]|T) <= \alpha$) then
        Discard T* as candidate clean version of T
    else if ($P(T^*[i]) >= \gamma * P(T)$) then
        Keep T* as candidate clean version of T
    else
        Discard T* as candidate clean version of T
    end if
end for
```

All the candidate replacement versions which have a $P(T^*|T)$ lower than a fixed threshold α are put into 'rejected' category and are not considered as alternative rectifications. The value of α is set low enough to avoid adding noisy, infrequent tuples (which are likely dirty tuples, and not clean outliers) from being stored in the probabilistic database. All T^* with $P(T^*|T)$ values higher than another fixed threshold β are placed into an 'accepted' category and are included in the Probabilistic database without further computation. The value of β is set high to have a high bar for clean T^*s.

Tuples having $P(T^*|T)$ lying between α and β are kept in 'processing' category and decisions on those tuples are made based upon the ratio of prior values of candidate and original tuples, $P(T^*)/P(T)$. Using a ratio avoids the problem of using a fixed threshold that is oblivious to the prior of the tuple. It is more likely that a frequent tuple is corrupted than a rare tuple but BayesWipe could not give the appropriate probability value due to the absence of error statistics. That is why candidate clean rectification T* in this category, which has significantly higher prior than the prior of the original tuples, are stored in probabilistic database. Also, tuples for which both the original and the candidate tuples have a low prior will be allowed in the probabilistic database since it is more likely that they are clean outliers. On the other hand, if the candidate clean tuple has a low prior value, but the original tuple is much more frequent, it is likely that the T^* is a dirty one. Tuples in this category are, therefore, only allowed in the probabilistic database if this ratio is larger than a threshold γ. The values of α, β and γ are set empirically. The observed best values for α, β and γ are 0.009, 0.5 and 5 respectively.

Chapter 5

Experimental Setup and Results

5.1 Experimental Setup

In order to evaluate BayesWipe-PDB against BayesWipe-DET, all experiments are performed in the same environment in which BayesWipe performs data cleaning.

Data Set

Experiments are performed on same dataset that BayesWipe[15] uses in order to clean data. This is a used car dataset which is crawled from Google Base. Similarly, for BayesWipe data cleaning, only the attributes which are categorical in nature are picked. resulting in 8 attributes. This observed database is clean. Then, I introduced noise in attributes randomly. I follow the same algorithm to introduce a noise as that of BayesWipe data cleaning. Experiments are performed on dataset whose size varies from 1000 to 30000 tuples. Synthetic noise is introduced to dataset which varies from 1% to 25%.

Random queries

In order to perform the evaluation on the information extracted by queries from BayesWipe-PDC dataset and BayesWipe-DDC dataset, all the queries are selected at random. Queries are selected on various columns of database for a particular domain value of the column. Particular column and corresponding domain value for a query is selected at random.

Hardware and Software

To perform the experiments, a Dell Optiplex Machine is used. This machine has 64 bits Windows 7 operating system and runs with 8 GB of random access memory and Intel Core2 Quad CPU Q9650. MystiQ [8], a prototype of Probabilistic database management system is used to provide query processing functionality on probabilistic database. Since MystiQ uses standard database system to store the data, SQL Server [22] is used to store probabilistic as well as deterministic data. All the evaluation experiments are implemented in Java.

5.2 Experiments and Results

BayesWipe-PDB is experimentally evaluated against BayesWipe-DET in two scenarios. In first scenario, BayesWipe-PDB is a probabilistic database which stores all alternative rectifications generated by BayesWipe for a dirty data tuple. Multiple random queries are run against BayesWipe-PDB and BayesWipe-DET and their result sets are compared with result set from ground truth to evaluate their effectiveness. Figure 5.1 shows the comparison of recall values for multiple random queries on the database. For a fix threshold 0.01, recall value of the result set from BayesWipe-PDB is higher than the recall value of BayesWipe-DET in every case since more numbers of relevant results are returned from bigger BayesWipe-PDB.

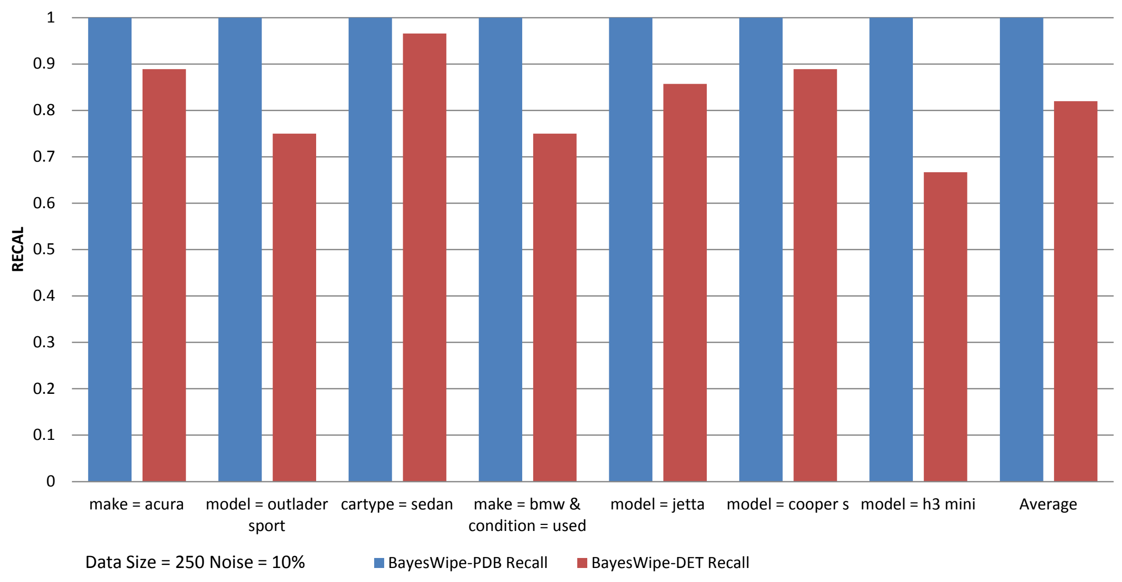


Figure 5.1: Recall of queries run against BayesWipe-PDB and BayesWipe-DET

Figure 5.2 shows the precision of same queries. It is quite evident that precision of the results goes down significantly. BayesWipe-PDB retain many irrelevant results since it stores all alternative rectifications.

Average precision and recall does not give a good enough idea about the comparison of BayesWipe-PDB and BayesWipe-DET when multiple random queries are used to generate the results set. So, instead of that, I use two non-normalized metrics, overall loss and overall gain, to show the comparison of BayesWipe-PDB and BayesWipe-DET.

Overall loss and gain are calculated for 100 randoms queries over BayesWipe-PDB and BayesWipe-DET. Overall loss is defined as the total increase in the number of false positives when BayesWipe-PDB is used instead of BayesWipe-DET. Overall gain is defined as the increase in

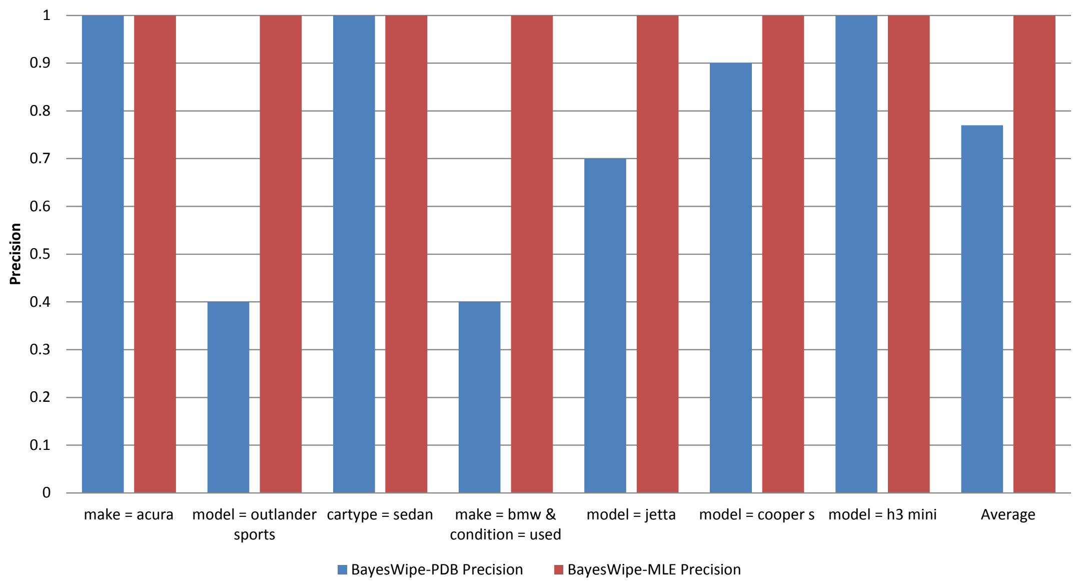


Figure 5.2: Precision of queries run against BayesWipe-PDB and BayesWipe-DET

number of true positives when all alternative rectifications are considered. Figure 5.3 shows the overall gain and overall loss.

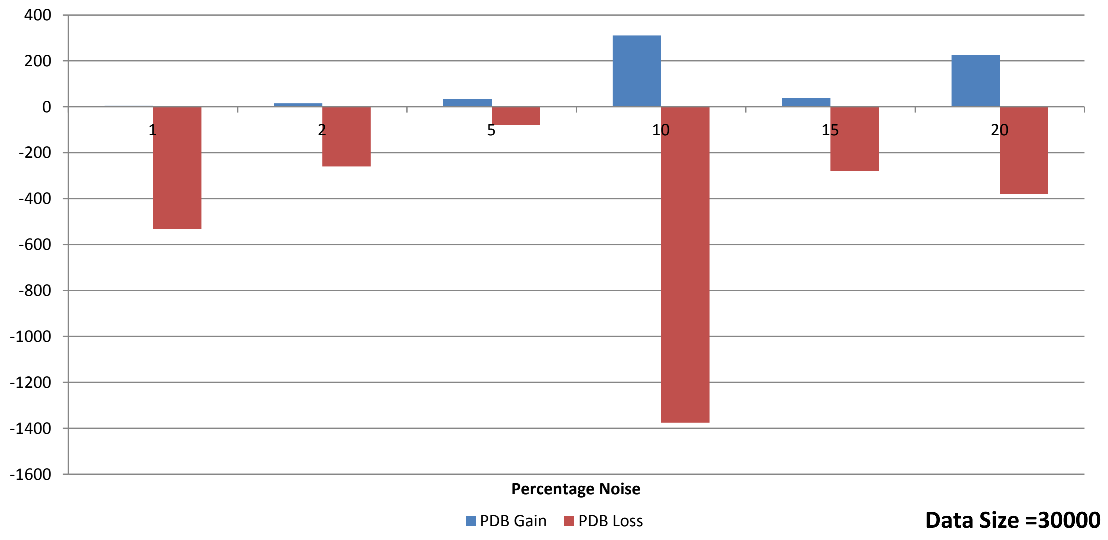


Figure 5.3: Average Gain in true positive and average Loss in false positive – BayesWipe-PDB vs BayesWipe-DET

As shown by figure 5.3, keeping multiple alternatives in a probabilistic database definitely increases the numbers of true positive results generated but increase in false positive results is significantly higher which can not be ignored.

BayesWipe-PDB is also compared against BayesWipe-DET on scalability. For that, average query processing time and size of database in terms of numbers of tuples of BayesWipe-PDB is compared against average query processing time and size of database of BayesWipe-DET.

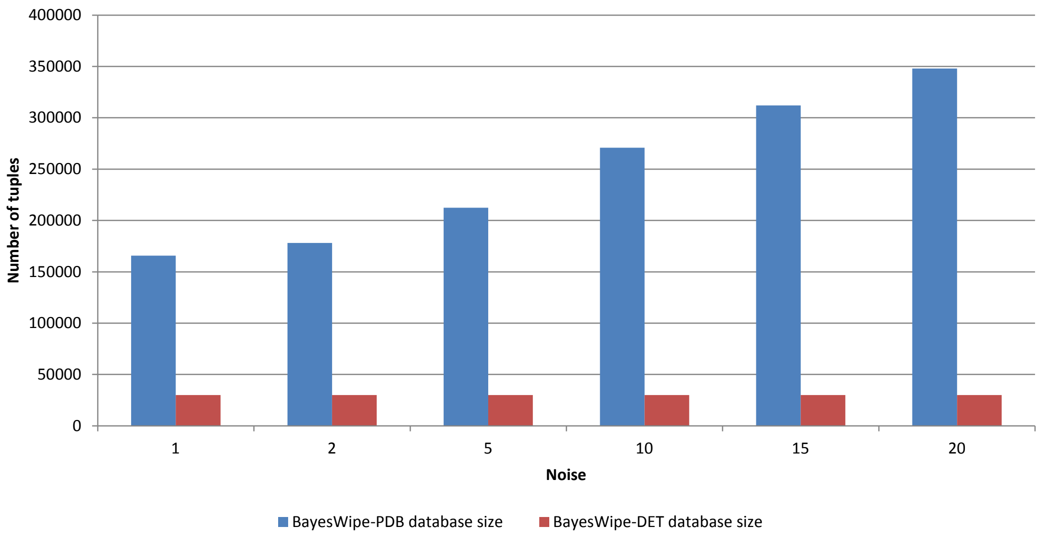


Figure 5.4: BayesWipe-PDB size against BayesWipe-DET size

Figure 5.4 shows the database size comparison of BayesWipe-PDB and BayesWipe-DET. It is quite evident that the size of BayeWipe-PDB will be larger than BayesWipe-DET. This experiment is done in order to check what is the worst case size that can be obtained and if the change in size is manageable or not. As the noise in dirty data increases, the size of BayesWipe-PDB, a probabilistic database generated as the result of keeping all alternatives rectifications in BayesWipe process, also increases. Similarly, the average query processing time(averaged over 100 random queries) of BayesWipe-PDB is much worse than the BayeWipe-DET. This is accredited to two reasons. First, the size of underlying database is hugely different in both approaches. Second, there is not industry strength probabilistic database management system available. Figure 5.5 shows the comparison of average query processing time.

Figure 5.6 shows the impact of threshold value on the accuracy of query results from BayesWipe-PDB. x-axis represents the threshold value where as y-axis respresents the precision/recall value. As the value of threshold increase, it eliminates more number of query results

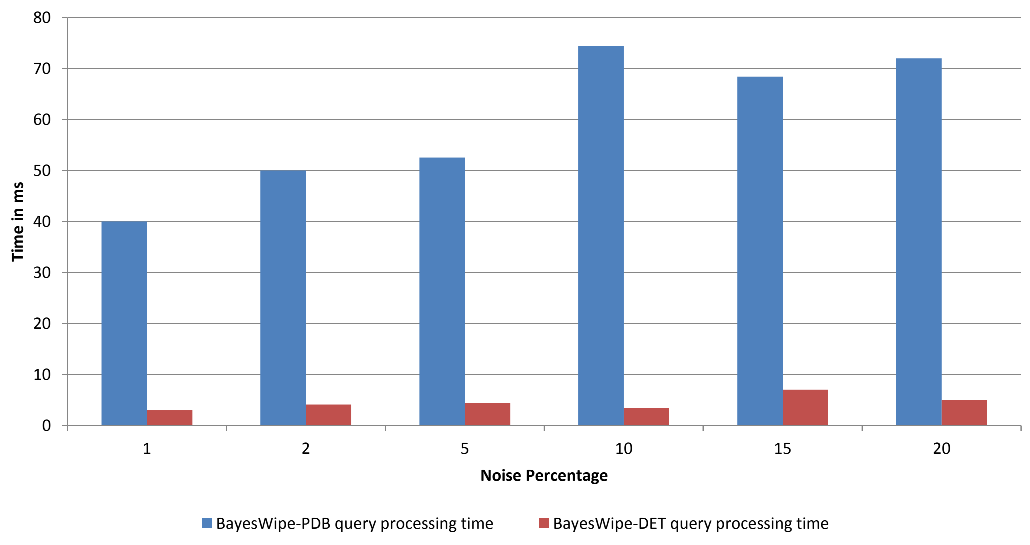


Figure 5.5: Average query time comparison BayesWipe-PDB and BayesWipe-DET

which will increase the precision but it will impact the recall value of result set. Graph shows how average precision and average recall varies with threshold value. Data set, used here, has 30000 tuple and has 5% of noise level. Average of precision and recall is taken over 10 random queries.

Thresholding is applied after the execution of query over probabilistic database, so it does not make any improvement in physical space requirement and query processing time.

In second scenario BayesWipe-PDB is optimized with pre-pruning. BayesWipe-PDB does not contain all alternative rectifications for a dirty data tuple, instead a pruning algorithm is used to generate a subset of candidate replacement tuples that are produced by BayesWipe as alternative rectifications for a dirty data tuple. This probabilistic database is compared against BayesWipe-DET for usefulness of the query results as well as on scalability. Figure 5.7 shows the gain in true positive results and loss in false positive results generated when random queries are executed against BayesWipe-PDB with pre-pruning and BayesWipe-DET. This graph shows that in every case, the numbers of false positives generated in the process of getting more true results is far less than the gain in number of true positive results.

Figure 5.8 compares the overall gain and Loss of BayesWipe-PDB with pruning and without pruning.

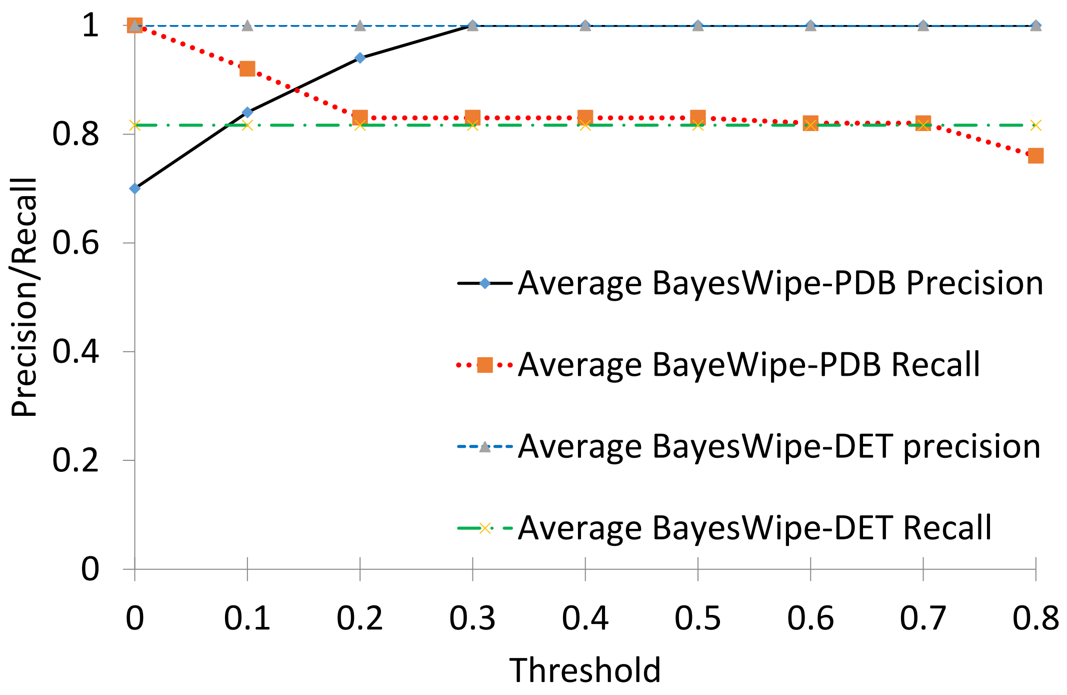


Figure 5.6: Precision and recall of BayesWipe-PDB with thresholding vs precision and recall of BAyesWipe-DET

BayesWipe-PDB with pruning is evaluated for scalability similarly to BayesWipe-PDB without pruning. BayeWipe-PDB with pruning makes a significant improvement in query processing time and the size of the probabilistic database stored as compared to BayesWipe-PDB without pruning. Figure 5.9 shows the processing time of BayesWipe-PDB without pruning, BayesWipe-PDB with pruning and BayesWipe-DET.

Figure 5.10 shows the database size of BayesWipe-PDB without pruning, BayesWipe-PDB with pre-pruning and BayesWipe-DET.

5.3 Result Summary

In these experiments, BayesWipe-PDB is compared against BayesWipe-DET. BayesWipe-PDB is evaluated against BayesWipe-DET in two separate modes.

In first mode, BayesWipe-PDB is created from outcomes of BayesWipe data cleaning system by giving equal importance to all alternative rectifications of a dirty data tuple and all of them is stored into a probabilistic database. In this evaluation, it is found that, although querying

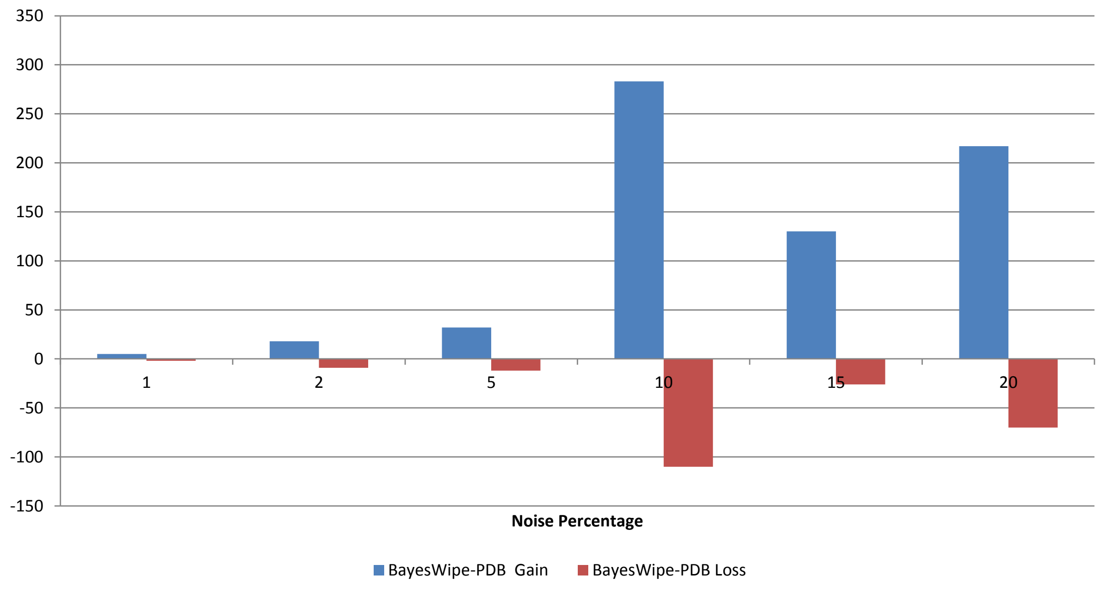


Figure 5.7: Average Gain and Loss from BayesWipe-DET when BayesWipe-PDB with pruning is used

BayesWipe-PDB, produces more true positive results than BayesWipe-DET, but in lure of producing more true results, BayesWipe-PDB also produces more false positive results. It is observed that total average gain in true positive results is very small as compared to the loss in false positive results. An appropriate value of threshold can help in improving the accuracy of the results by eliminating some of the irrelevant results but it will also affect the recall.

In second mode, BayesWipe is generated by not giving equal importance to all the alternative rectifications. It considers only selected rectifications and prunes many unnecessary candidate replacement tuples produced by BayesWipe data cleaning. Evaluation of this mode shows significant improvement in loss caused by more numbers of false positive without affecting the gain in true positive. Following this approach, the gain in true positive results is higher than loss in false positive. This approach also makes a significant improvement in size of probabilistic database without losing any valuable information. The improvement in space is up to 90% in the size of probabilistic data. It also improves the query processing time significantly. The experimental results summary is shown in figure 5.11

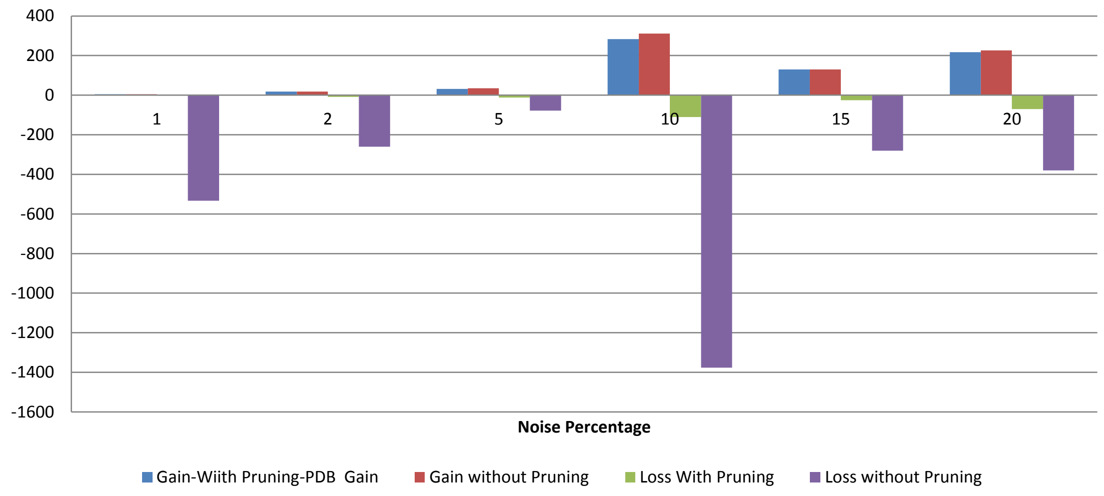


Figure 5.8: Average Gain and Loss from BayesWipe-DET when BayesWipe-PDB with pruning and BayesWipe-PDB without pruning is used

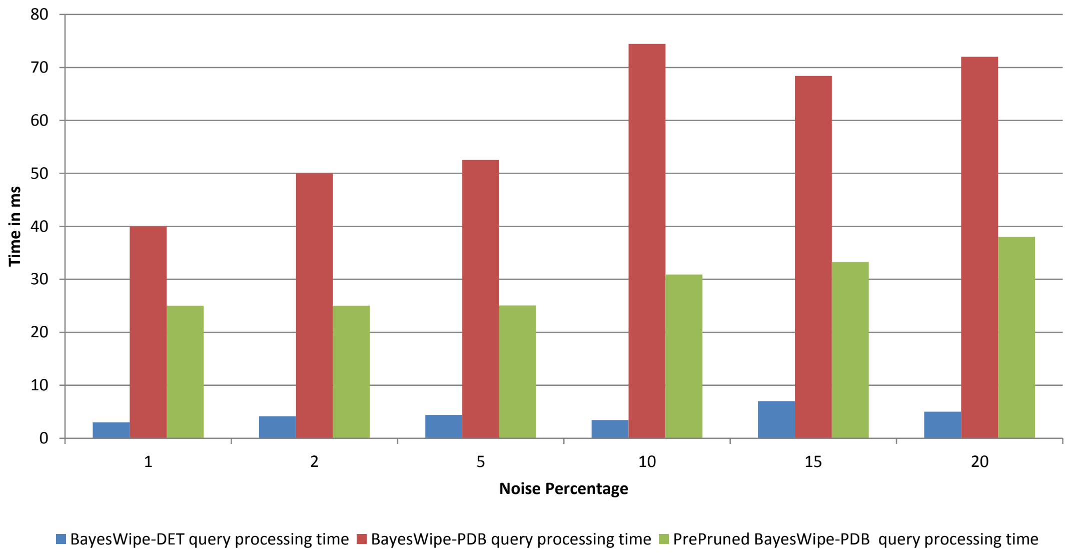


Figure 5.9: Average query processing time of BayesWipe-DET, BayesWipe-PDB, BayesWipe-PDB with pre-pruning

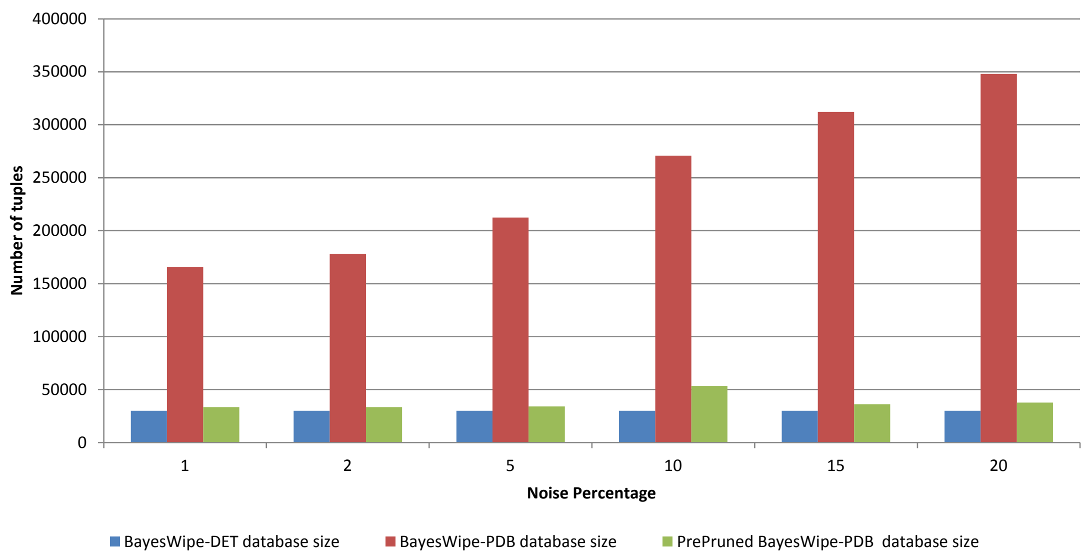
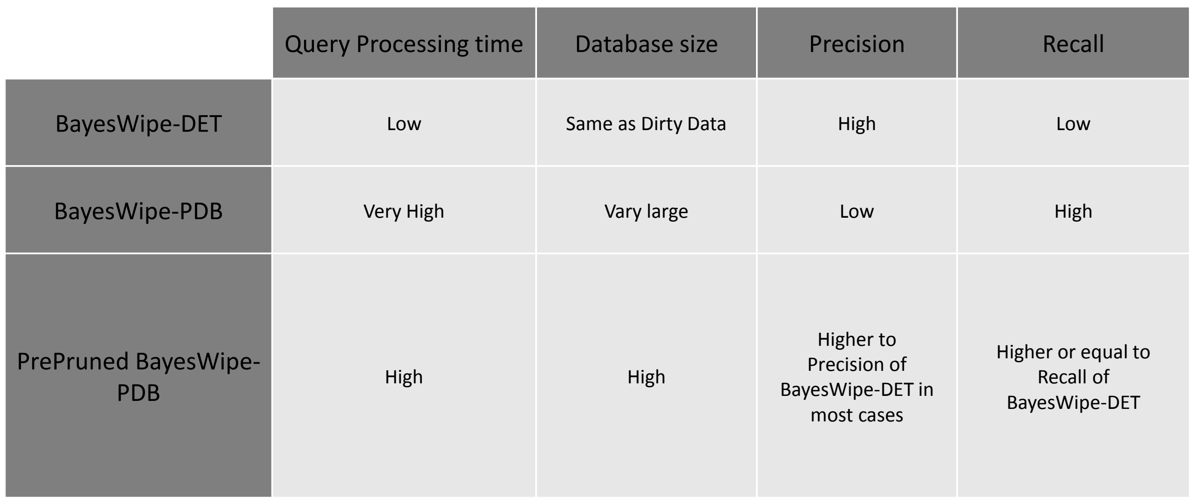


Figure 5.10: Comparison of probabilistic database size with BayesWipe-PDB with pre-pruning and BayesWipe-PDB without pre-pruning



	Query Processing time	Database size	Precision	Recall
BayesWipe-DET	Low	Same as Dirty Data	High	Low
BayesWipe-PDB	Very High	Vary large	Low	High
PrePruned BayesWipe-PDB	High	High	Higher to Precision of BayesWipe-DET in most cases	Higher or equal to Recall of BayesWipe-DET

Figure 5.11: Experimental results summary

Chapter 6

Conclusion

In this thesis, I investigated the trade-offs of retaining multiple alternative rectifications of the dirty data instance in data cleaning process against keeping only the most likely rectification of the dirty data instance. The investigation is done in the context of a data cleaning system, called BayesWipe. I demonstrated the method of considering multiple alternative rectifications by retaining all rectifications in a probabilistic database. I compared the two different outcomes of BayesWipe data cleaning system – BayesWipe-DET deterministic outcome that retains only one rectification per dirty data tuple and BayesWipe-PDB, a probabilistic outcome that considers more than one alternative rectifications per dirty data tuple. I compared them on scalability and usefulness of the query results obtained from both outcomes. I showed, how keeping more alternatives would help in generating more true results than the true result generated from BayesWipe-DET. I also pointed out a few potential issues related to scalability and usefulness of the query results associated with the approach that keeps all alternative rectifications.

Further, I presented how thresholding can improve the usefulness of the query results by eliminating irrelevant results which have probability values lower than the fixed threshold value from the result set. I showed how user can controll he precision/recall of query results by varying the value of threshold. In addition to this, I presented a novel pre-pruning technique in order to avoid too many low probable rectifications to be stored in the probabilistic database. I showed, how pre-pruning helped in both reducing the database size as well as improving the query time without giving anything significant on usefulness of the query results.

In my thesis, I presented how keeping multiple alternative rectifications from data cleaning rather than one rectification could give you some benefits in terms of getting more true positive results but this approach comes with a high query processing time and a larger physical space requirement. So, one should consider all his/her objectives before opting for this alternative approach.

REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 586–597. VLDB Endowment, 2002.

[2] C. R. Association. Challenges and opportunities with big data. http://cra.org/ccc/docs/init/bigdatawhitepaper.pdf, 2012.

[3] L. E. Bertossi, S. Kolahi, and L. V. S. Lakshmanan. Data cleaning and query answering with matching dependencies and matching functions. In *International Conference on Database Theory*, 2011.

[4] G. Beskales. Modeling and querying uncertainty in data cleaning, 2012.

[5] G. Beskales, M. A. Soliman, I. F. Ilyas, and S. Ben-David. Modeling and querying possible repairs in duplicate detection. *Proceedings of the VLDB Endowment*, 2(1):598–609, 2009.

[6] G. Beskales, M. A. Soliman, I. F. Ilyas, S. Ben-David, and Y. Kim. Probclean: A probabilistic duplicate detection system. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 1193–1196. IEEE, 2010.

[7] P. Bohannon, W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for data cleaning. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 746–755. IEEE, 2007.

[8] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. Mystiq: a system for finding more answers by using probabilities. In *SIGMOD*, pages 891–893, 2005.

[9] R. Bryant, R. H. Katz, and E. D. Lazowska. Big-data computing: Creating revolutionary breakthroughs in commerce, science and society, 2008.

[10] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 551–562. ACM, 2003.

[11] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *Knowledge and Data Engineering, IEEE Transactions on*, 16(9):1112–1127, 2004.

[12] W. Fan, F. Geerts, L. Lakshmanan, and M. Xiong. Discovering conditional functional dependencies. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. Ieee, 2009.

[13] I. Fellegi and D. Holt. A systematic approach to automatic edit and imputation. *Journal of the American Statistical association*, pages 17–35, 1976.

[14] A. Hartemink. Banjo: Bayesian network inference with java objects. Available: http://www.cs.duke.edu/ amink/software/banjo.

[15] Y. Hu, S. De, Y. Chen, and S. Kambhampati. Bayesian data cleaning for web data. *arXiv preprint arXiv:1204.3677*, 2012.

[16] T. Imieliński and W. Lipski Jr. Incomplete information in relational databases. *Journal of the ACM (JACM)*, 31(4):761–791, 1984.

[17] E. Knorr, R. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3):237–253, 2000.

[18] I. Lazaridis and S. Mehrotra. Approximate selection queries over imprecise data. In *Data Engineering, 2004. Proceedings. 20th International Conference on*, pages 140–151. IEEE, 2004.

[19] H. Liu, S. Shah, and W. Jiang. On-line outlier detection and data cleaning. *Computers & chemical engineering*, 2004.

[20] T. Minka, W. J.M., J. Guiver, and D. Knowles. Infer.NET 2.4, 2010. Microsoft Research Cambridge. http://research.microsoft.com/infernet.

[21] E. Rahm and H. H. Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[22] J. R. Shapiro. *Microsoft SQL Server 2005: the complete reference*. McGraw-Hill, 2007.

[23] D. Suciu, D. Olteanu, C. Ré, and C. Koch. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180, 2011.

[24] H. Xiong, G. Pandey, M. Steinbach, and V. Kumar. Enhancing data analysis with noise removal. *Knowledge and Data Engineering, IEEE Transactions on*, 2006.