Cost-Conscious Cleaning of Massive RFID Data Sets*

Hector Gonzalez Jiawei Han Xuehua Shen
Department of Computer Science, University of Illinois at Urbana-Champaign
{hagonzal, hanj, xshen}@uiuc.edu

Abstract

Efficient and accurate data cleaning is an essential task for the successful deployment of RFID systems. Although important advances have been made in tag detection rates, it is still common to see a large number of lost readings due to radio frequency (RF) interference and tag-reader configurations. Existing cleaning techniques have focused on the development of accurate methods that work well under a wide set of conditions, but have disregarded the very high cost of cleaning in a real application that may have thousands of readers and millions of tags. In this paper, we propose a cleaning framework that takes an RFID data set and a collection of cleaning methods, with associated costs, and induces a cleaning plan *that optimizes the overall* accuracy-adjusted cleaning costs *by determining the conditions under which inexpensive methods are appropriate, and those under which more expensive methods are absolutely necessary.*

1 Introduction

Radio frequency identification (RFID) technology has seen increasing adoption rates in applications that range from supply chain management, asset tracking, to monitoring of pharmaceutical products. The success of these applications depends heavily on the quality of the data stream generated by the RFID readers, *i.e.*, the reader should detect all the tags that are present within its read range, and should not detect tags that are not present, or that are present but due to business rules should not be detected. But the reliability of current RFID systems is far from optimal. Data cleaning is thus essential for the correct interpretation and analysis of RFID data, but given the enormous volume of information, diverse sources of error, and rapid response requirements, it can be a challenging task.

The problem of cost-conscious data cleaning is defined as follows. Given three inputs: (1) a set of tag readings, which form a representative sample of the possible set of readings, each labeled with the correct location for the tag, and also annotated with contextual information, such as item type (metal, water contents), area conditions, or tag protocol; (2) a set of cleaning methods with associated per-tuple cleaning costs; and (3) a per-tuple missclassification cost, which may be constant, or a function of the tag reading and incorrectly assigned location, our goal is to learn a *cleaning plan* that *identifies the conditions (feature values) under which a specific cleaning method or a sequence of cleaning methods should be applied in order to minimize the expected cleaning costs, including error costs.*

In this paper we introduce a framework for the cost-conscious cleaning of RFID data streams, a novel data cleaning approach capable of adapting to the diversity of conditions found in a large RFID implementation, and able to clean large data sets at a minimum cost with high accuracy. We summarize our contribution as follows:

A cost optimization view of RFID data cleaning: We present a novel formulation of data cleaning as a cost minimization problem. This view allows us to balance accuracy and cleaning costs in order to correctly clean large number of tag readings with minimum resources.

The introduction of the cleaning plan: In general finding the optimal assignment of cleaning methods to tag reading conditions, such that cost is minimized, is hard. We introduce the cleaning plan, *a decision tree, induced by greedily choosing features that provide the highest* expected cost reduction *as an approximation that works well under a variety of conditions.*

An effective cleaning method based on Dynamic Bayesian Networks (DBNs): We propose a new cleaning method dynamically adjusts the belief state (probability that the tag is present) given the last observation. Our extensive experiments show that under some conditions DBN-based cleaning can outperform or complement methods based on smoothing windows.

*The work was supported in part by the U.S. National Science Foundation NSF IIS-05-13678 and NSF BDI-05-15813.

The rest of the paper is organized as follows. Section 2 presents the structure of RFID data sets. Section 3 presents a detailed treatment of cost-conscious cleaning and develops the notion of cleaning plan. Section 4 reports experimental and performance results. We discuss related work in Section 5 and conclude our study in Section 6.

2 RFID Data

The operation of an RFID system generates a stream of data resulting from *interrogation cycles* occurring at recurring time intervals at each Reader. The data generated for each *interrogation cycle* is usually a set of tuples of the form $(EPC, Reader, time)$. The tuple can be augmented with extra information, such as the *antenna* used by the Reader, the *power level* of the interrogation signal, or the *tag type* (class 0, class 1, generation 2, *etc.*). From the application perspective it is possible to look at multiple *interrogation cycles* as a single unit known as a *read cycle* in such case we get tuples of the form $(EPC, Reader, time, responses)$, where *responses* is the number of *interrogation cycles* when the tag identified by EPC was correctly inventoried.

3 Cost-Conscious Cleaning

Cleaning of large RFID data sets can be an expensive problem. It requires very fast processing when real time or near real time responses to tag location queries are required. Existing work on RFID cleaning has mainly focused on improving the accuracy of a stand-alone technique and largely ignored costs.

A cost-conscious approach to cleaning is to determine the context under which inexpensive methods work well, and try to clean as many RFID data as possible with such methods, while applying more expensive techniques only when absolutely necessary. Figure 1 presents the architecture of the cleaning framework. We have a set of labeled data, which contain instances of tag readings annotated with features that describe the context in which the reading took place, and labeled with the cleaning methods that classify each case correctly. There is also a repository of available methods with cost information. The cleaning engine induces a cleaning plan that minimizes the cost of data cleaning while maintaining high accuracy, and uses this cleaning plan to clean the RFID stream.

Example Table 1 presents an example of labeled tag readings, and the cleaning results of applying three methods, *fix_1* which is a fixed-window method with window size 1, *dbn* which is a DBN-based method, and *pat* is a pattern matching method. Each method makes a prediction of 1 (tag present), 0 (tag absent); the *label* column is the correct prediction.

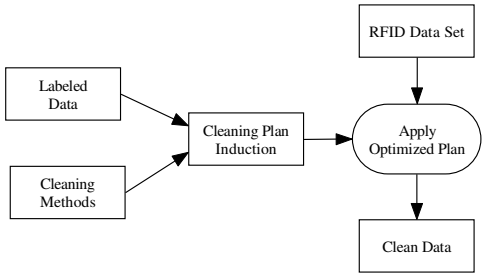

Figure 1: Architecture of the cleaning framework

cycle	tag	reader	metal	fix_1	dbn	pat	label
1	1	shelf	no	1	0	0	1
1	2	shelf	no	0	0	1	0
1	3	shelf	yes	0	1	0	1
2	4	shelf	no	0	0	0	0
2	5	shelf	yes	0	1	0	1
2	6	shelf	no	0	1	0	1
3	7	door	yes	1	1	0	0
3	8	door	no	1	1	0	0
3	9	door	yes	1	1	1	1

Table 1: Labeled tag cases

3.1 Cleaning Methods

Definition 3.1 *A cleaning method is a classifier M that takes as input, a* tag case*, which is a tuple of the form $(\langle EPC, t\rangle, \langle f_1, f_2, ..., f_k\rangle)$, where each f_i is a feature describing one characteristic of the tag identified by EPC or its environment at time t, and makes a prediction of the form $(\langle EPC, t\rangle : loc, conf)$, where loc is the predicted location for the tag identified by EPC at time t, and $conf$ is the confidence that the classifier has in the prediction.*

Cost model for a cleaning method. The cost of a cleaning method consists of the per-tuple cleaning cost and the error cost. The per-tuple cleaning cost is a function of two variables: (1) the amortized per tuple training cost, and (2) the cost, in term of storage space and running time, for labeling a tag reading. The error cost is what we have to pay for each misclassified tag reading. The error cost can be a scalar value that simply penalizes issuing an incorrect tag location by a constant or a matrix.

Features available to a cleaning method. The context in which tag readings take place defines the feature space. Intuitively, features are important information regarding a tag at at the time of a reading. Features can be classified into four groups: (1) *Tag features*, which describe characteristics of the tag, such as communications protocol, vendor, price, or his-

tory of recent tag detections, (2) *Reader features*, which describe the reader, including the number of antennas, protocol, price, and vendor, (3) *Location features*, which describe the location where the reading took place, including the type of area being monitored (*e.g.*, door, shelf, and conveyor belt), or the sources of interference in the area, and (4) *Item features*, that describe the item to which the tag is attached, including item composition (*e.g.*, water or metal content), physical dimensions, or if it is a container.

A DBN-based view of cleaning

In this section we propose a new cleaning method based on Dynamic Bayesian Networks (DBNs). We assume there is a hidden process that determines the true location of a tag, but we only see noisy observations of the hidden process. The model maintains a *belief state* that is the probability that the tag is present or not at a reader given the past observations. DBN-based cleaning has the advantage that it does not require us to remember recent tag readings, and as opposed to window smoothing, it gives more weight to recent readings (even within a window) than older ones.

A simple implementation of the model is to define a single hidden variable X_t that is true if the tag is present at the reader's location at time t, and a single observation variable e_t, which is a noisy signal dependent on X_t. We can compute the most likely current state X_{t+1} given the observations $e_{1:t+1}$ as:

$$P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) P(x_t|e_{1:t}) \quad (1)$$

where $P(e_{t+1}|X_{t+1})$ is the known probability of observing a certain reading given a true state of the world, $P(X_{t+1}|x_t)$ is the known probability of changing from one true state to another, and $P(x_t|e_{1:t})$ is our previous belief state. The observation and transition models can be learned from the data or be given by the user.

3.2 Cleaning plan

A *cleaning plan* specifies the conditions under which we should apply each cleaning method in order to minimize the expected total cost of cleaning a data set. A natural way to model a cleaning plan is to use a decision tree induced on a labeled set of tag cases. Each node in the tree can be split along the distinct values of a feature. The tag cases in each node share the same values on the features used to split the nodes from the root of the tree to the node. Leaves in the tree represent subsets of data that will be labeled using the same data cleaning strategy.

Figure 2 presents a cleaning plan induced on the label data set from Table 1. This plan divides the data set into three groups, each with a different cleaning strategy.

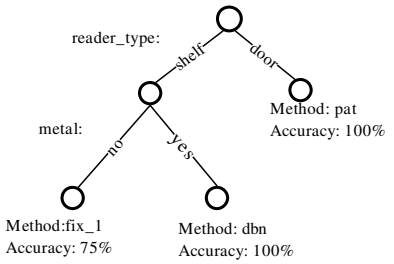

Figure 2: Example Cleaning Plan

3.2.1 Node cleaning costs

In order to decide how to construct the cleaning plan, we need a way to evaluate the expected cost to clean the tag cases residing at a node. This requires us to determine the optimal sequence of cleaning methods that should be applied to the cases in the node such that total cleaning cost is minimal.

A *cleaning sequence* is an ordered application of cleaning methods to a data set. The cost of applying a *cleaning sequence* $S_{D,M} = M_{s_1} \rightarrow M_{s_2} \rightarrow ... \rightarrow M_{s_k}$ to a data set D, denoted as $C(S_{D,M})$, is the sum of the cost of applying M_{s_1} to the complete data set D, applying M_{s_2} to the cases that M_{s_1} classified incorrectly, applying M_{s_3} to the cases that both M_{s_1} and M_{s_2} classified incorrectly, and so on, until applying M_{s_k} to the cases that all the other methods failed to classify, and the error cost incurred on the cases that no cleaning method is able to classify correctly. At cleaning time we can use the confidence provided by the cleaning as a threshold to determine if another method should be applied.

Definition 3.2 ***Optimal cleaning sequence****. Given a set of labeled tag cases D, and a set of cleaning methods $M = \{M_1, ..., M_k\}$, the* optimal cleaning sequence *to clean D with M, denoted as $S^*_{D,M}$, is the ordered sequence of methods from M with minimal cost.*

Determining the *optimal cleaning sequence* for a node, when we have non-terminal cleaning methods, is hard in practice, as methods can be correlated, *e.g.*, they may be good or bad at classifying the same cases, and confidences do not necessarily match classification correctness. In general, it is possible for high confidence predictions to provide wrong labels, and vice versa. We can use a greedy algorithm to choose an approximation to the optimal cleaning sequence as follows: (1) sort the cleaning methods on ascending order of *accuracy-adjusted cleaning costs*, and select the method with the lowest cost, (2) remove from D the tuples correctly classified by the method selected in Step 1, and (3) repeat the process until a terminal method

is selected, or when the methods or cases are run out. We will denote the greedy approximation to the optimal method sequence as $\hat{S}^*_{D,M}$.

3.2.2 Node splitting criteria

When deciding the structure of the cleaning plan, we need a criterion to determine when to split a node and the feature to choose for the split. This decision will be based on the difference in cleaning costs of the cases in the node before and after the split. We call this criteria *expected cost reduction*, defined as follows.

Definition 3.3 ***Expected cost reduction.*** *Given a set of cleaning methods M and a data set D that can be split using feature f, with $|f|$ distinct values, into the subsets $D_1, \ldots, D_{|f|}$, we define the* expected cost reduction *of the split as,*

$$C(\hat{S}^*_{D,M}) - \sum_{i=1}^{|f|} \frac{|D_i|}{|D|} \times C(\hat{S}^*_{D_i,M}) \qquad (2)$$

The cleaning plan can be constructed using a typical Top Down Induction of Decision Trees (TDIDT) algorithm, such as [6], using Eq. (2) as the node splitting criterion.

4 Experimental Evaluation

In this section we present a thorough analysis of the performance of the *cleaning plan* on several data sets and compare its cost and accuracy with a number of cleaning methods applied individually. All the experiments were conducted on an AMD Athlon XP 1.2 GHz System with 1GB of RAM. The system ran cygwin 1.5.20-1 and gcc 3.4.4.

Data Synthesis

The data sets for our experiments were generated by a synthetic RFID data generator that simulates the operation of RFID readers under a wide variety of conditions. The simulation is composed of two components. The first simulates the movement of items (associated with tags) through predefined paths. We control tag behavior by altering speed, tag communication protocol, density, and item characteristics. The second component simulates tag detection by a number of RFID readers with varying tag detection rates that depend on factors such as tag characteristics, reader quality, RF noise, and distance to tags.

Our experiments compare the performance of a cleaning plan, denoted as *plan*, with several cleaning methods used independently. Performance is measured in average per-tuple cleaning costs and accuracy, which is the percentage of correctly cleaned records. The first method is our proposed DBN-based cleaning, denoted as *DBN*. The second method is the implementation of variable-window smoothing [4], denoted as *var*. The third method uses fixed-window smoothing, where three window sizes, 1, 2, and 3, are chosen and denoted as *fix_x*. We also implemented two user-defined rules, one denoted as *maj*, which resolves multi-reader detection conflicts by majority voting, and *pat* which is used to detects only bell shaped detection signals at doors. We use a cost model that assigns per tuple classification cost to each model that is proportional to the computational requirements, with fixed window methods being the cheapest, and pattern recognition methods the most expensive.

Cleaning performance for a diverse reader/tag setup. In this experiment we compare the performance of the cleaning methods when applied to a data set obtained from a diverse configuration of readers and tags. This configuration is a small sample of the reader setups that may be observed at a large RFID implementation. We simulate readers operating in low and high noise environments, with static and moving tags located at varying distances form readers. Figure 3 presents the expected (from training data) and the actual (from testing data) per tuple classification costs, and the accuracy of each method. We see that *plan* is cheaper and more accurate that competing methods, this is due to its ability to correctly match methods to readers.

Cleaning plan performance for a changing reader/tag setup. In this experiment we analyze the way the cleaning plan adapts to changing RFID configurations to outperform any individual cleaning method. We start with an easy configuration with low noise and high detection rates, and progressively increase complexity by introducing moving tags, and interference. In Figures 4 and 5, we see that *plan* correctly adapts to the changing conditions outperforming individual methods.

Cleaning plan performance for different noise levels. Figures 6 and 7 present the cleaning results on a system that contains difficult to detect tags attached to items containing water or metal, and easy to detect tags. *plan* again is the winner, and by looking at the cleaning plan, we see that the induction algorithm correctly associated cheap methods with easy tags, and more expensive methods to hard to detect tags.

Figure 8 shows that our cleaning plan induction algorithm scales nicely with a growing training data set sizes, and with a growing number of cleaning methods.

5 Related Work

Several smoothing methods have been proposed for cleaning of RFID streams, [1] proposes the fixed-window smoothing. [4] proposes a variable-window smoothing that adapts window size dynamically ac-

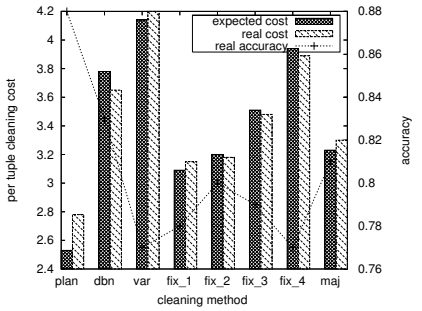

Figure 3: Cleaning performance for a complex setup

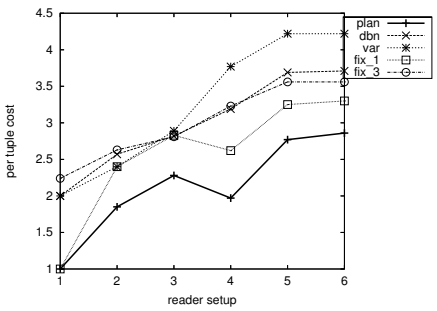

Figure 4: Cleaning costs vs. Setup complexity

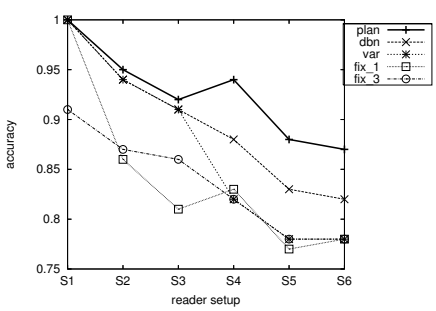

Figure 5: Cleaning Accuracy vs. setup complexity

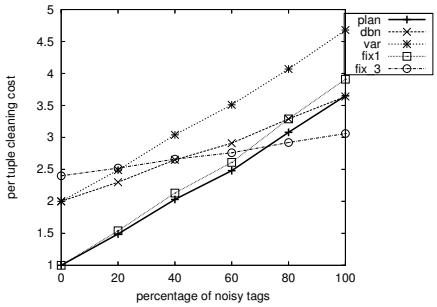

Figure 6: Cleaning costs vs. Percentage of noisy tags

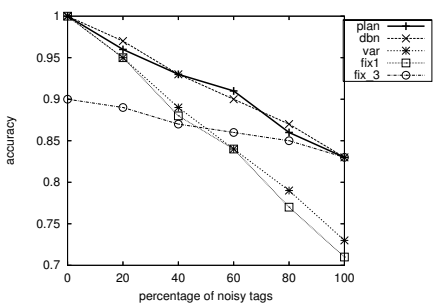

Figure 7: Cleaning Accuracy vs. Percentage of noisy tags

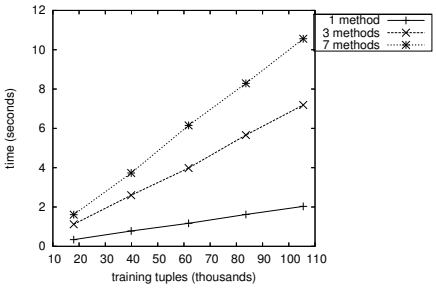

Figure 8: Runtime vs. Training data set size

cording to tag detection rates. [5] presents a framework to clean RFID data streams by applying a series of filters. Our work differs from these studies in that the cost-conscious cleaning framework operates at a higher level than any single technique and learns how to apply multiple techniques to increase accuracy and reduce costs. We also take advantage of previous studies on management and mining of RFID data sets [3, 2] to incorporate important contextual information into the cleaning process and take it as a source of labeled data.

6 Conclusions

We have proposed a novel cost-conscious framework for cleaning RFID data sets that learns when and how to apply different cleaning techniques in order to optimize total cleaning costs and accuracy. The key intuition behind the framework is to determine the situations in which inexpensive cleaning methods work well, while only applying expensive methods when really necessary. We proposed an efficient cleaning plan induction method based on the idea of top-down induction of decision trees and the novel concept of cleaning cost reduction. Moreover, our DBN-based cleaning method takes tag readings as noisy observations of a hidden state and performs effective data cleaning.

References

[1] C. Floerkemeier and M. Lampe. Issues with rfid usage in ubiquitous computing applications. In *In Pervasive Computing (PERVASIVE) Lecture notes in Compute Science*.

[2] H. Gonzalez, J. Han, and X. Li. Flowcube: Constructuing RFID flowcubes for multi-dimensional analysis of commodity flows. In *Proc. 2006 Int. Conf. Very Large Data Bases (VLDB'06)*, Seoul, Korea, Sept. 2006.

[3] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and analysis of massive RFID data sets. In *Proc. 2006 Int. Conf. Data Engineering (ICDE'06)*, Atlanta, Georgia, April 2006.

[4] S. R. Jeffery, M. Garofalakis, and M. J. Franklin. Adaptive cleaning for RFID data streams. In *Proc. 2006 Int. Conf. Very Large Data Bases (VLDB'06)*, Seoul, Korea, Sept. 2006.

[5] S. R. Jeffrey, G. Alonso, M. Franklin, W. Hong, and J. Widom. A pipelined framework for online cleaning of sensor data streams. In *Proc. 2006 Int. Conf. Data Engineering (ICDE'06)*, Atlanta, Georgia, April 2006.

[6] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.