

An Intelligent Approach to Handle False-Positive Radio Frequency Identification Anomalies

Authors: Peter DARCY (Corresponding Author), Bela STANTIC, and Abdul SATTAR

INSTITUTE FOR INTEGRATED AND INTELLIGENT SYSTEMS, GRIFFITH UNIVERSITY,

QUEENSLAND, AUSTRALIA, TEL: +617 555 28502, FAX: +617 555 28066

EMAIL: {P.DARCY, B.STANTIC, A.SATTAR}@GRIFFITH.EDU.AU

Abstract

Radio Frequency Identification (RFID) technology allows wireless interaction between tagged objects and readers to automatically identify large groups of items. This technology is widely accepted in a number of application domains, however, it suffers from data anomalies such as false-positive observations. Existing methods, such as manual tools, user specified rules and filtering algorithms, lack the automation and intelligence to effectively remove ambiguous false-positive readings. In this paper, we propose a methodology which incorporates a highly intelligent feature set definition utilised in conjunction with various state-of-the-art classifying techniques to correctly determine if a reading flagged as a potential false-positive anomaly should be discarded. Through experimental study we have shown that our approach cleans highly ambiguous false-positive observational data effectively. We have also discovered that the Non-Monotonic Reasoning classifier obtained the highest cleaning rate when handling false-positive RFID readings.

Keywords: Radio Frequency Identification - RFID; False-Positive Anomaly; Non-Monotonic Reasoning; Bayesian Network; Neural Network.

1 Introduction

Radio Frequency Identification, commonly abbreviated to RFID, is an automated means to wirelessly track tagged items. It utilises a unique identification tag which is interrogated by a reader and, therefore, it is able to capture the information of where and when the tags appear within a certain location. The wide array of applications that could employ this technology range from a shopping cart trolley which tallies up the groceries of the user automatically to a constantly tracking postal service to find exact locations of customers' parcels. Although the hardware of RFID systems has advanced to the point of technology effectively tracking items seamlessly, the RFID systems still suffer from several afflictions that prevent it from becoming widely accepted by the commercial sector.

One of the serious problems that prevents RFID systems from obtaining high levels of commercial attention is the erroneous data it produces, in particular, false-positive readings. Past research has utilised different methodologies such as manual cleaning, deferred rule cleansing and filtration techniques but no methodology has succeeded in completely eliminating false-positive anomalies thus far. These methodologies suffer from a lack of automation, intelligence and advanced analysis to properly correct false-positive readings without inadvertently introducing artificial anomalies from the cleaning process. The most problematic situation occurs with ambiguous readings when false positive anomalies cannot be easily cleaned.

To correct these anomalies, we propose in this paper a concept that utilises an advanced analysis

coupled with intelligent classifying algorithms to eliminate false-positive readings within the RFID data warehouse. The methodology first evaluates each tag within the data set to discover a suspicious observation and key characteristics used in future processes. The observation along with the characteristics are then passed to one of three classifier algorithms to determine whether the flagged reading should continue to be stored or deleted. We have chosen three intelligent classifiers to determine whether the readings should be deleted. These are: Bayesian Network, Neural Network and Non-Monotonic Reasoning Logic Engine.

We conducted four different experimentations to find the highest achieving configurations of the techniques and the overall best performing classifier. Each of the first three experiments investigated the most effective training algorithm/formula for each of the three classifiers. From this experimental evaluation, we found that genetic training algorithms with both 10 and 100 chromosomes performed the best for both the Bayesian and Neural Networks respectively and that the π formula allowed the Non-Monotonic Reasoning Logic Engines to obtain the highest cleaning rate. The fourth experiment tested each of the three classifiers to determine which cleaned the highest amount of false positive anomalies. We could not compare our approach to any of the previously proposed concepts presented in literature because those methods lack automation or the ability to correct ambiguous observations. From this final experiment, we found that the Non-Monotonic Reasoning classifier achieved the highest performing cleaning rate.

The remainder of this paper has been organised as following: Section 2 will examine the background of our concept and Section 3 will provide an analysis of related and previous work already conducted in this field of study. We propose our methodology and scenario for experimentation in Section 4. The details, results and analysis of the experimentation will be given in Section 5 followed directly by our conclusion and future work suggestions in Section 6.

2 Background

RFID systems have been prominent within the research community since 1948 when Harry Stockman published the first academic paper to feature this technology [27]. This is due to the fact that there are massive benefits available in a wide array of commercial applications from the integration of RFID systems. Unfortunately, due to problems, including ambiguous false positive data anomalies, the potential uses have only minutely been brought into reality.

2.1 RFID

Radio Frequency Identification (RFID) is the seamless transmission of identification tags to readers in an effort to locate objects with ease. It utilises tags that are interrogated by a reader to transmit their unique identifier which is passed via middleware to a data warehouse [7]. A diagram illustrating the flow of data between these objects may be viewed in Figure 1. The tags fall into three categories: the active in which tags rely solely on batteries to transmit their identifier at great distances; the semi active in which a battery is harnessed to extend only the range of the transmission; and the passive tags in which energy is scavenged from the electro-magnetic pulse generated by the reader to transmit the unique identifier. We have chosen to focus on passive tags as they are the most commercially viable being both cheap and having an eternal lifespan due to the lack of a need for battery power.

Unfortunately, the benefits of the passive RFID tag have been limited by the flaws within the practical implementation of the device. Specifically, there are four issues preventing the wide scale adoption of the system: the data is low level [17]; the incoming data is prone to error [16]; large volumes of data are generated over relatively small amounts of time[24]; and there is an increasingly

complex nature of the spatial-temporal readings [29].

Low level data is hazardous to the application in that the resulting observations are meaningless without being put into context. Without higher level event transformations, the data will not be able to be utilised beyond basic operations. The issue of large volumes of data also hinders the applications as the generated tag readings flood the system with incoming data. It is reported that Wal*Mart stores, which has integrated RFID into its daily routines, records several TeraBytes of data daily [12]. As the data is being recorded, there are also problems in that the recordings may consist of duplicate, missed and wrong readings which mislead the data analysts. Additionally, when the data is recorded there are certain issues that arise due to the complex nature of the spatial-temporal readings especially when employing readers that are not stationary.

Within the erroneous issue of the tags, the two persistent anomalies are false positive readings in which data is captured where it was not meant to and false negative readings in which data is not captured where it is required to be. False positive readings consist of two main scenarios in which a recording has either been duplicated by mistake [5], or read at a reader where the physical tag has not actually been present [1]. The cause for such occurrences usually amounts to tags being captured outside the reader range due to the environmental setup or where more than one reader is observing the same tag due to overlapping interrogation zones. It has been stated that the filtration of these unwanted data records are major flaws within all RFID applications [12] [13].

2.2 Bayesian Networks

A Bayesian Network is a probabilistic classifier utilising probabilistic values which are given to different variables to arrive at the most likely conclusion. To find the likelihood of each outcome,

the variables of each conclusion are multiplied. The highest achieving outcome is then identified as the likeliest candidate to be the correct conclusion [21]. The equation utilised for each conclusion finds the probability of the entire case by deriving all the probabilities of observations within the statement. The network is built upon these probabilities and, as such, needs guidance as to what each of these values should be. To train the probabilities, a genetic algorithm may be utilised by reducing each of the probabilities into genes of a chromosome and finding the optimal configuration [15].

The chromosomes are initiated by setting the genes to small random values. After a fitness test to derive the highest performing chromosomes of the population, a certain amount of this population is destroyed and the resulting fittest chromosomes are crossed-over to re-establish the original population number. The cross-over function consists of the genes of the two parents being traded to create children of the parents. Common types of cross-overs include one point, two-point or uniform cross over. Eventually, a mutation occurs on a small percentage of the resulting population to avoid local minima issues.

2.3 Neural Networks

As seen in Figure 2, an Artificial Neural Network is a classifier designed to emulate the learning behaviour of the brain. It does this by creating a fixed amount of Neurons which are trained to deliver a certain output when fed various input. The entire process has actually been based on the biological neuron pictured in Figure 3. Dendrites will receive information which is passed to the cell body whose objective is to pass the information into the axon when certain requirements are met and, thus, to dendrites of other neurons via the synapse connection. The crucial difference between

a digital Neuron, shown in Figure 4 and its biological counterpart is that there is a computational limit to the amount of hidden units that may be present within a network. Unfortunately, technology has not advanced enough to effectively and efficiently emulate the amount of Neurons the human brain possesses, which is estimated to be between 10 billion and 1 trillion [30].

Input: *Neuron output*

Output: *Activation function output*

if *The value is greater than 0* **then**

| *Set output to +1.*

else

| *Set output to -1.*

end

Algorithm 1: The algorithm for the Hard Limiter Activation Function utilised within a Neural Network.

The Artificial Neural Network consists of three main layers: the *Input Layer*, *Hidden Layer(s)* and the *Output Layer* [20]. The processes include receiving inputs which are modified at a central sum area. The Neuron will then apply an activation function such as the hard limiter or sigmoidal functions, which are displayed in Algorithm 1 and Equation 1 respectively, to derive a value for the output. With regards to training the network, there are several techniques available such as the *Back-propagation* [26], [4] and *Genetic Algorithms* [14], [25] which are both considered as leaders with regards configuring Artificial Neural Networks.

$$f(x) = 1/(1 + \exp(-x)) \tag{1}$$

When attempting to configure the Neural Network weights, one method that exists is to utilise training algorithms. Two dominant training algorithms that have been proven to excel in net-

work training are the Back-Propagation Algorithm and the Evolutionary Neural Network. Back-Propagation relies on the concept of training the network by propagating error back through the network via modifying the weights after the output has been calculated [26]. The algorithm uses either a predetermined limited amount of iterations or the Root-Mean-Square error threshold of the calculated output as stopping criteria [4]. A flow diagram interpretation of how the Back-Propagation Algorithm operated may be viewed in Figure 5.

The Evolutionary Neural Network training algorithm in contrast to Back-Propagation utilises the theory of genetic evolution to train the network weights. Similar to the genetic algorithm process of training a Bayesian Network, all the weights are added into a chromosome as genes to be manipulated according to the fittest output obtained [14]. The weights are initialised as small random numbers which are checked for either obtaining a high enough score or if the amount of generation limit has been reached. In the case that neither of the stopping criteria has been met, the algorithm will examine each chromosome in relation to achieving the correct output. A certain amount of the unfit chromosomes are then destroyed within the population to be replaced with child chromosomes of two of the fitter chromosomes. The child chromosomes can be created with different means such as one-point, two-point or uniform crossover [25]. Mutation will then be applied to a certain small percent of the population to assure that the network avoids problems such as network paralysis or local minima [6]. A general flow diagram interpretation of the Evolutionary Neural Network training process may viewed in Figure 6.

2.4 Non-Monotonic Reasoning

Non-Monotonic Reasoning is a distinct logic that operates by having the ability to make various deductions which will then be eliminated until the conclusion has been drawn from various observations. What sets it apart from classic theories of logic is its ability to arrive at various solutions as opposed to just one over-reaching conclusion. Clausal Defeasible Logic (CDL) has been designed to incorporate the Non-Monotonic Reasoning logic precisely within a computational environment and allows it to be run within various applications [2]. The direct benefit to embedding Clausal Defeasible Logic is that applications will be able to deterministically arrive at intelligent conclusions when given various input parameters.

An example in which Non-Monotonic Reasoning would decipher the correct solution is in relation to Siberian Huskies barking [18]. It is generally accepted that dogs bark, however, the Siberian Huskies are dogs which do not bark. This has been represented within the logical engine map pictured in Figure 7 where the rule that the Siberian Huskies do not bark outweighs the rule in which dogs bark. In most cases, the logic declares a general rule which will then be defeated by specific rules that defy the previously declared rule.

To logically interpret this, it must first be stated as a fact that Siberian Huskies are dogs. After this fact has been established, we can state that it is plausibly correct to assume that all dogs bark. At the last step, we state that Siberian Huskies plausibly do not bark and that this latter rule defeats the former. To properly illustrate any ambiguous problem within the Clausal Defeasible Logic framework, the Decisive Programming Language (DPL) must be utilised [3]. Using the DPL, the Siberian Huskies problem translates into the following statements:

$$\text{Siberian_Huskies}(x) \rightarrow \text{Dogs}(x).$$

R1: $\text{Dogs}(x) \Rightarrow \text{Bark}(x)$.

R2: $\text{Siberian_Huskies}(x) \Rightarrow \sim\text{Bark}(x)$.

R2 > R1

After this code has been written, it is passed into the CDL compiler which transforms it into an implementable C function which will return the logical conclusions for any given inputs. The aspect that helps CDL surpass other logical reasoning aspects is its ability to utilise different strengths of ambiguity when it determines its conclusion. These levels of ambiguity strength are represented using the following formulae:

- μ : The strongest of the algorithms as it will only respond to factual information to arrive at a conclusion.
- α : An algorithm in which any conjunction of π and β are used to reach the conclusion.
- π : The algorithm in which ambiguity is propagated to reach its conclusion.
- β : The algorithm in which ambiguity will not be allowed to be used to draw its conclusions.
- δ : The formula in which the disjunction of π and β are used to draw conclusions.

3 Related Work

With regards to previous work, there are three main methodologies used to eliminate the false positive readings within the RFID data set. The first method is the manual cleaning tool in which data sets are cleaned by the user identifying and discarding anomalies [22]. The manual cleaning

tool is not specifically designed to be utilised with RFID data sets, but it may be applied to the observational data warehouse. The second methodology is the automated use of rules defined by the user which is applied at a deferred stage of the cleaning process [23]. The final method is an automated filter that corrects all on incoming data with use of a dynamic smoothing window [16].

The manual cleaning method is effective but costly in terms of time and effort on the user's part. An automated solution would be more practical particularly for mass amounts of data. The rule-based approach is automated, however has problems when cleaning ambiguous information due to the fact that not all cases, in reality, are solvable with a general rule. While filtration, also an automated solution, suffers from the analytical data available which is only incoming data, therefore the scope of data to analyse is very limited. Additionally, false-positive anomaly correction within RFID systems is similar to the Data Mining of Imbalanced Data Sets [8]. However, due to the unique nature of the observational data, the classifiers have to be tailored to handle cases that arise specifically within captured RFID data.

4 Methodology

To properly illustrate how our methodology performs, we will first review the motivation and architecture of the program. This includes the Feature Set Definition, Bayesian Network, Neural Network, Non-Monotonic Reasoning and Loading phases. Next, we will provide details of the ideal scenario to design our system most effectively. Finally, any assumptions we have made to ensure that the algorithms operates correctly will be listed.

4.1 Motivation

We created this system with the intended goal of correcting ambiguous false positive RFID data anomalies. To accomplish this, we created an intelligent analytical feature set definition that ties directly into three classifiers to determine the correct course of action. We chose to integrate classifiers as the current related work (such as rule-based solutions) lack the intelligence needed to handle ambiguous anomalies. This would allow our methodology to be able to counteract the anomalous data in an intelligent and automated manner. This results in our methodology overcoming the current problems that exists in current state-of-the-art techniques and provide a cleaning process that obtains both effective and efficient superiority over existing methodologies. We chose the three classifiers as opposed to other techniques due to the Bayesian Network [10], Neural Network [11] and Non-Monotonic Reasoning [9] have all been able to effectively correct false-negative RFID anomalies in the past. We also chose Bayesian and Neural Networks as probabilistic methods to be compared with the deterministic Non-Monotonic approach.

A sample of the ambiguous scenario we attempt to correct is shown in Table 1. The table consists of the three vital RFID observational data: the tag's identifier-EPC (Electronic Product Code), the reader's identifier and the time of the event. As seen in the table, the tag with an EPC of T1 is seen at Location R3, R4, R2, R4, R1 and R1 where the third observation would be flagged as a suspicious reading. In previous methodologies, this flagged observation would automatically be deleted due to the proximity of the readers in the second and fourth observations. However, using the 'Map Data' our methodology may make a more intelligent cleaning decision.

The map data, as seen in Figure 8, is used to determine validity of the observation. Within the figure, the lines that house the circle represent walls, the large circles represent the reader

range, the letters inside the circle identify the reader and the lines between the circles represent two readers being within proximity of each other. Knowing this map data, our methodology provides a higher intelligence when it attempts to clean by examining the first and fifth observations where the locations R3 and R1 are read. Knowing that both locations in the first and fifth observation are geographically within proximity to the suspicious reading, it will make a judgment that the reading is actually valid. This will then lead onto the fourth observation which it will discover as a false-positive anomaly due to the second, third, fifth and sixth readings not being within the proximity of the fourth which in this example scenario, is correct. We believe that by enhancing the rules with our concept will apply much needed intelligence in scenarios where the obvious action will not clean the data set.

4.2 Architecture

As it may be observed in Figure 9, the design of our system has been broken into three sections: the *Feature Set Definition Phase*; the *Classifier Phase*; and the *Modification Phase*. The Feature Set Definition Phase is the first process that is conducted within our application in which the raw data is searched and sorted to find suspicious readings and the circumstances surrounding each of these observations. The Classification is where the system deviates between three different classifiers, the *Bayesian Network*, *Neural Network* or *Non-Monotonic Reasoning Engines*. Each classifier has one goal which is to determine if the flagged reading should be deleted or kept within the data set. This decision is based solely upon the input gathered from the *Feature Set Definition Phase*. After each classifier has determined the validity of the observation, it will then pass the decision onto the *Modification Phase* which will either delete or keep the value being passed to it. This entire process

may be viewed in the high level pseudo-code interpretation found in Algorithm 2.

Input: *Raw Observational Data Set*

Output: *Modified Observational Data Set*

foreach *Reading* **do**

if *The observation is flagged for Suspicion* **then**

 | *Find the readings needed for Feature Set Definition.*

 | *Analyse the readings for key characteristics.*

 | *Pass the characteristics and reading to Classifier. if The classifier finds the value*

 | *to be kept then*

 | *Do not modify the data set.*

 | **else**

 | *Delete the flagged observation.*

 | **end**

end

end

if *The user chose to replace the data set* **then**

 | *Delete the Original Data Set. Save the Modified Data Set at Original Data Set*

 | *location.*

else

 | *Save the Modified Data Set.*

end

Algorithm 2: A high level pseudo-coded version of the algorithm used within our methodology.

4.2.1 Feature Set Definition Phase

The first stage of the program is the *Feature Set Definition Phase* whose main goal is to analyse the data to discover suspicious readings and investigate key characteristics surrounding the flagged observation. Initially, this phase breaks the tag readings into Tag Streams (Definition ??) designed to analyse the route of one tag. We define Tag Streams as individually analysed streams for one tag from the mass amount of readings. A tag will be flagged suspicious if the difference in timestamps exceed the user-defined duration it should take to reach the location, or if the geographical locations of the readers are not within proximity. To determine the geographic validity of the readers the program utilises a table named 'Map Data', which is constructed by the user and reflects the geographic layout of all adjacent readers within the static environment. As illustrated in Figure 10, there are five observational values that are ascertained: a , b , x , c and d . The values of observations a and b are the reading taken two readings or one reading respectively before the suspicious reading x . The c and d readings are the observations which been recorded once and twice after the suspicious reading. From all these observations, the timestamp and the location are all recorded and used in further analysis.

After the values of a , b , x , c and d with their respective timestamps and locations have been found, the Feature Set Definition Phase further investigates key characteristics of the data. The characteristics are comprised of ten different binary mathematical operations, however additional characteristics maybe be added by the user. Each of the characteristics contain spatial and temporal information regarding the observations before and after the suspicious readings. With regards to the proximity of timestamps, we have utilised the value of half a second. This time value may be altered to better suit the application for which it is designed. The characteristics we discover are

as follows:

- $b.loc \leftrightarrow x.loc$
- $c.loc \leftrightarrow x.loc$
- $b.time == x.time$
- $c.time == x.time$
- $b.loc == x.loc$
- $c.loc == x.loc$
- $a.loc \leftrightarrow x.loc$
- $d.loc \leftrightarrow x.loc$
- $b.time \leftrightarrow x.time$
- $c.time \leftrightarrow x.time$

The data used in these characteristics are five different values that have two sub-values each. The five values include the observations of a , b , x , c , and d , which each have the time (time) and location (loc) for each value stored. The relations our methodology uses in analysis include when values are within certain proximity which is represented as \leftrightarrow , or are equivalent which is represented as $==$. It is important to note that the function that states that the two values are within proximity of each other have different meaning between the location and time. With regards to location, the proximity is determined by the 'Map Data' table whereas the proximity with regards to time refers to the time value of the two observations being within the user defined time value

of each other. After all these characteristics have been gathered, they are passed onto the various classifying methodologies as inputs to determine whether or not the flagged item should remain within the data set.

4.2.2 Classifier Phase - Bayesian Network

The first option that the *Classifier Phase* can utilise is the Bayesian Network. In this example we have considered the Bayesian Network to have ten inputs that correspond to the analytical characteristics found at the end of the *Feature Set Definition Phase*. Using these ten inputs the Bayesian Network will then determine, based on the weights it has obtained through training, whether the flagged observational reading should be kept within the database, see Equation 2. This will result in one output known as the *keep_value* which will be set to either true or false. An example of the tabular structure of the network may be viewed in Table 2. This *keep_value* output will be passed to the *Modification Phase* at the end of this process at which point the entire application will repeat each time a suspicious reading is encountered. We also set all binary input numbers from 0 and 1 to 0.1 and 0.9 respectively to allow for higher mathematical functions to benefit from avoiding to multiply by zero.

$$P(\textit{keep_value}) = \prod_{i=1}^n P(\textit{b.loc} \leftrightarrow \textit{x.loc}, \textit{c.loc} \leftrightarrow \textit{x.loc}, \quad (2)$$

$$\textit{b.time} == \textit{x.time}, \textit{c.time} == \textit{x.time}, \textit{b.loc} == \textit{x.loc},$$

$$\textit{c.loc} == \textit{x.loc}, \textit{a.loc} \leftrightarrow \textit{x.loc}, \textit{d.loc} \leftrightarrow \textit{x.loc},$$

$$\textit{b.time} \leftrightarrow \textit{x.time}, \textit{c.time} \leftrightarrow \textit{x.time})$$

We have chosen a Genetic Algorithm to train the Bayesian Network weights based on the various

test cases that may arise. The Genetic Algorithm will have the population of the chromosomes to determine the ideal number of chromosomes utilised for training purposes. The mutation rate of the Genetic algorithm being utilised will be 1% for the top 10% chromosomes with regards to fitness and 5% for all other chromosomes. After the best weight configuration has been determined, the network will be utilised to compare it against the Neural Network and Non-Monotonic Reasoning approaches.

4.2.3 Classifier Phase - Neural Network

An Artificial Neural Network (ANN) is the second option we have chosen for the Classifier Phase which utilises weighted neurons to determine the validity of the flagged value. Like the Bayesian Network, this ANN will use the ten inputs gathered from the *Feature Set Definition Phase* to pass through the network and obtain one output. This network configuration has been displayed graphically in Figure 11. The network shall be comprised of a single hidden layer with eleven hidden nodes resulting in 121 weights between all the nodes. We specifically wanted to choose more hidden units than inputs and only one layer as we have found that multi-layered networks do not necessarily enhance the performance of the classifier.

Input: *None*

Output: *Trained Weights*

Set weights to small random numbers.

while *There are still Iterations* **do**

| *Calculate the Output.*

| *Check the amount of iterations and RMS Error.*

| **if** *The amount of Iterations or RMS Errors has been reached* **then**

| | *Break the while loop.*

| **end**

| *Adjust the weights based on propagated error.*

end

Output the weights and RMS Error.

Algorithm 3: The algorithm for the Back-Propagation training we used in our methodology.

We have also set the momentum and learning rates to 0.4 and 0.6 respectively and have utilised a Sigmoidal Activation Function. Additionally, as with the Bayesian Network we shall use the numbers 0.1 and 0.9 rather than the binary numbers of 0 and 1 respectively. Two prominent training algorithms have been utilised to properly configure the Neural Network. The first is the Back-Propagation Algorithm while the second is the Genetic Algorithm which has also been utilised within the Bayesian Network. Both algorithms will use a limited amount of iterations as stopping criteria for the training. The pseudo-coded version of the Back-Propagation and Genetic Algorithms may be viewed in Algorithms 3 and 4 respectively. Additionally, various amounts of chromosomes will be utilised for the Genetic Algorithm.

Input: *None*

Output: *Trained Weights*

Initialise the Population

```
while There are still Generations do
| Run the selection process.
| Check the stopping criteria.
| if Ideal fitness has been reached then
| | Break the while loop.
| end
| Cross over selected chromosomes.
| Mutate random amount of chromosomes.
end
```

Output winning Weights and Fitness.

Algorithm 4: The algorithm for the Genetic Algorithm training we used in our methodology.

4.2.4 Classifier Phase - Non-Monotonic Reasoning

The final classifier we have utilised within our implementation is Non-Monotonic Reasoning Logic Engines. The actual algorithm utilises a series of rules that we have created based upon the input analysis variables obtained from the *Feature Set Definition Phase*. From this, the logic engines determine the correct course of action to either keep the value or not based on the different levels of ambiguity we enforce.

The rules utilised within the logic engine may be examined in Table 3. The four symbols that are used to interact with the values within the rules are the logic AND operator \wedge , the negative operator \sim , the equal operator $==$ and our use of the double arrow \leftrightarrow to illustrate proximity

between the two analysis variables. The process used to find the conclusion from the lookup table created by the logic engine can be seen in pseudo-code in Algorithm 5.

Input: *DPL Rules, Analytical Values, Formula*

Output: *Lookup table, Conclusions*

Input DPL Rules into CDL.

Store lookup table for later use.

Open up lookup table.

while *There are still entries in lookup table* **do**

if *Found the entry for given Inputs* **then**

Find the conclusion associated with Inputs.

Break the while loop.

end

end

Output conclusion found.

Algorithm 5: The algorithm for the Non-Monotonic Reasoning lookup process used in our methodology.

As a default case where neither `keep_val` or `~keep_val` are encountered, the logic engine will keep the flagged reading to avoid artificially introduced false negative observations. Additionally, the order in which they have been written in this document is also the order of priority with regards to finding the conclusion.

4.2.5 Modification Phase

After each intelligent classifier has determined whether or not to keep or delete the flagged reading, it will pass it to the *Modification Phase*. After the decision has been received, the application

will then delete the identified value in the original data warehouse. The system will also only flag potential anomalies rather than automatically delete the flagged items if the user would like to examine the observations prior to modification.

5 Experimental Results and Analysis

In order to investigate the applicability of our concepts, we conducted four experiments. The first three were dedicated to finding the optimal configuration of each classifier whereas the last focused on the comparison of the three classifiers. In this section, we describe both the scenario, database structure, assumptions and environment in which we conducted these experiments and an analysis of the experiments. Furthermore, we describe the experimental evaluation and present the results of four experiments which we conducted. The first three are designed to determine the highest achieving configuration of the Bayesian Network, Neural Network and Non-Monotonic Reasoning classifiers. In the fourth experiment, the highest achieving classifiers have been compared against each other to find which one achieves the highest cleaning rate.

5.1 Scenario

The ideal scenario which we envisioned for our application would be an enclosed RFID static environment. It is important that such an environment exists as the program requires knowledge of the geographical landscape with regards to the locations of the readers. The motivation for such an application came from the idea of a building that houses important items which need to be monitored thoroughly such as a hospital or an elderly care unit. Although the scenario has been designed specifically to integrate with an RFID environment for their specific applications,

this software may be utilised for an array of domains that utilise spatial and temporal coordinates within the data set. Due to medical facilities already employing RFID systems in this manner [28], we believe that this scenario is particularly realistic to be beneficial to similar applications that could be conducted.

5.2 Database Structure

To store the observational data within a data warehouse we have designed a database structure modelled after the Data Model for RFID Applications (DMRA) [19]. DMRA is currently being used within RFID middleware where it is designed to manage all RFID observational data efficiently. We have chosen only three tables from the many utilised in the application that are relevant to our methodology which houses information regarding the readers (Reader), tags (Object) and interaction between both of them (Observation). The structure of the tables used in this data warehouse is as follows:

- `READER (ReaderID, Name, Description)`
- `OBJECT (Epc, Name, Description)`
- `OBSERVATION (ReaderID, Value, Timestamp)`

Additionally, we have created another table 'Map Data', which stores information vital to our concept. The prime goal of this additional table is to store the readers which are within close proximity of each other. The structure for this additional table is as follows: `MAPDATA (ReaderID1, ReaderID2)`.

5.3 Assumptions

We have two prime assumptions for this application, first, that the environment is static and, secondly, that the locations of the readers are known. This is important as the concept relies heavily on both the mapped data to determine if a reading is suspicious and, subsequently, when it attempts to reason the validity of the observation if it has been flagged. This would require the application to be run in an RFID-enabled environment where the readers are mounted which would prevent it from changing locations. To create this pre-requisite condition, the user would have to first enter the map data into the application before the application has begun to correct anomalies. Additionally, any modifications to the environment or the reader locations would have to be updated within the map data table as well.

5.4 Environment

The code for this methodology was written and compiled in the C++ language utilising Microsoft Visual Studio C++ 6.0. The experiments were run on a Windows XP operating system running Service Pack 3. As outlined earlier, above, there are four main experiments which were conducted using the methodology. The first experiment was designed to test the highest performing Genetic Algorithm when training the Bayesian Network. For this experiment, the amount of chromosomes in the population was manipulated to find the highest performing number. The second experiment was designed to discover which training algorithm of either the Back-Propagation or Genetic Algorithm obtained the highest cleaning rate. For this experiment, the amount of chromosomes were modified and compared with the back-propagation algorithm to determine the highest achieving algorithm. The third experiment was designed to determine which formulae achieved the highest cleaning rate

within the Non-Monotonic Reasoning approach.

We specifically chose only to examine classifier techniques as the related work is not comparable due to either it not being able to clean ambiguous data or not using an automated process. The last experiment which was conducted took the highest achieving configurations of each of the classifiers and compared each methodology against the other. Four data sets were utilised for this experimentation, the first three were training sets in which 500, 1,000 and 5,000 scenarios were used to train the algorithms and find the optimal configuration. Each training set contained different scenarios to avoid the risk of over-fitting the classifiers.

The second data set was three testing sets in which 1,000, 5,000 and 10,000 randomly chosen scenarios were selected and passed to the application to have the anomalies eliminated. Each of these testing sets contained feature set definitions generated within our sample scenario. After each of the training and testing experiments have been conducted, the average of cleaning rate of the experiments has been derived for each technique and used to identify the highest achieving method.

5.5 Bayesian Network Experiment

For our first experiment, we conducted an investigation into the optimal amount of chromosomes which are needed to clean the false positive anomalies. To accomplish this, we created three Bayesian Networks that have been configured using a genetic algorithm with 10, 50 and 100 chromosomes. Each network was trained for 10 generations to breed and optimise the configuration. With regards to the set of data being used for training, we used 3 different “Training Cases” comprising 500, 1,000 and 5,000 false-positive anomaly scenarios. After these experiments were completed, the average of the 3 training cases was then extracted and, subsequently, used to determine the amount of

chromosomes that are needed to achieve the highest cleaning rate.

The results of this experiment are shown in Figure 12 with the Amount of Training Cases and Chromosomes are graphed against the Percentage of Correctness. It may be observed that the first viewable feature is the configuration that used 10 chromosomes to train the network obtained the highest average cleaning rate. As a result, the Bayesian Network using a Genetic Algorithm with 10 chromosomes will be utilised in the final experiment in which all three classifiers are compared. The lowest achieving configuration using 100 chromosomes tested upon 500 training cases was the Bayesian Network. The highest achieving configuration has been found to be the configurations with 10 and 50 chromosomes against 500 and 1,000 training cases respectively.

5.6 Neural Network Experiment

The second experiment we conducted was in relation to determining the highest performing network configuration for a Neural Network to clean anomalous RFID data. To do this, we trained the weights of the networks using the back-propagation (BP) and the genetic algorithms with 10 (GA-10), 50 (GA-50) and 100 (GA-100) chromosomes present. The performance of the resulting networks were determined based upon the correctness of the classification from 3 Training Cases using 500, 1,000 and 5,000 false-positive anomaly scenarios. Each configuration had been trained by 10 iterations or generations before the training experiment commenced. The main goal of this experiment was to determine the highest average achieving network trainer, thus the average of the three training cases has also been found.

From the experimental results shown in Figure 13, we have derived a general observation that the performance of the network is vastly improved with 50 and 100 chromosomes using the Genetic

Algorithm. The highest performing average of the Neural Network has been found to be the Genetic Algorithms when trained with both 50 and 100 chromosomes. As such, we decided to use the Genetic Algorithm with 100 chromosomes as the attempt to clean 500 training cases performed the highest. The lowest performing cleaning algorithm was the Back-Propagation algorithm when attempting to clean 1,000 training cases.

5.7 Non-Monotonic Reasoning Experiment

The main goal of the third experiment was to derive the highest performing Non-Monotonic Reasoning formula from the five different options used in Clausal Defeasible Logic. With this in mind, the μ (Mu), α (Alpha), π (Pi), β (Beta) and δ (Delta) formulae were each trained using 3 training cases containing 500, 1,000 and 5,000 false-positive scenarios each. Like the previous two experiments, the averages of each performing algorithm was ascertained and used to determine which of the five formulae would be utilised to proceed onto the final experiment. The results of this experiment are presented in Figure 14 where the observations have been structured with the Amount of Training Cases and the various Formulae have been graphed against the Percentage of Correctness.

From the results, it can be clearly seen that the highest performing formulae are μ , α and π . In contrast, the β and δ formulae both performed the least cleaning. With regards to the final experimentation, we have chosen the π formula as it performed the highest and is the most likely to continue to perform highly. The reasons as to why we rejected the μ and α formulae lie in the fact that the μ formula is strict in that it only accepts factual information and the α formula is connected directly to the β . Hence, we determined that the π formula would be superior to the other formulae.

5.8 Comparison Experiment

The goal of the final experiment was to determine which of the three highest performing classifier techniques would clean the highest percentage of a large amount of false-positive RFID anomalies. The three classifiers used in this experiment included the Bayesian Network trained by a Genetic Algorithm with 10 chromosomes (BN), the Neural Network trained by Genetic Algorithm with 100 chromosomes (NN) and the π of the Non-Monotonic Reasoning Engine (NMR). The classifiers were all chosen based upon the high performance found within the first three experiments previously discussed. Both of the Bayesian and Neural Networks had both been trained for 10 generations before these tests were conducted. As opposed to the previous experiments, we determined that three “Testing Cases” containing 1,000, 5,000 and 10,000 randomly chosen false-positive scenarios would be utilised to determine the highest performing classifier. To ascertain the highest performance, the average of each of the three test cases has been found from the results.

The results of this experiment is depicted in Figure 15 where the Amount of Test Cases and Classifier has been graphed against the Percentage of Correctness. From these results, it can be seen that the Non-Monotonic Reasoning Engine achieves the highest average cleaning rate among the other classifiers. The highest performing classifier has also been found to be the Non-Monotonic Reasoning when attempting to clean 1,000 test cases; whereas the lowest achieving classifier has been found to be the Neural Network when attempting to clean 10,000 test cases.

The Non-Monotonic Reasoning Engine outperformed the other classifiers in dealing with false-positive data due to the fact that it is a deterministic approach. The Bayesian and Neural Networks, by contrast, rely on a probabilistic nature to train their respective networks. The major drawbacks of this system are that it is specifically tailored for the static RFID cleaning problem, however, we

believe that the same concept may be applicable to any static spatio-temporal data enhancement case study. With regards to applying our methodology to other applications where the environment is dynamic, the Feature-Set Definition and Non-Monotonic Reasoning will need greater complexity to accommodate the change in anomalies. Although the test cases utilised in experimentation were small in comparison to the immense amount of RFID readings that get recorded in real world systems, we believe our methodology would behave similarly upon larger data sets.

6 Conclusions

In this work we have focused upon an automated and intelligent means to correct false positive RFID readings at a deferred stage of the capture cycle. The rationale behind our choice of creating such a methodology was to create a concept which will target ambiguous observations. We have shown through experimental evaluation that our presented concept obtains a high cleaning rate. Specifically, this work makes the following contributions to the field:

- We developed a new concept to clean ambiguous false-positive RFID observations.
- We proposed and utilised a highly intelligent and novel feature set definition technique to extrapolate the information needed to correctly act upon the false-positive anomalies.
- We introduced the use of Non-Monotonic Reasoning, Deferred Bayesian Network, and Artificial Neural Network classifiers coupled with our feature set definition to resolve ambiguous readings of RFID observations;
- We discovered that the Non-Monotonic Reasoning π formula performs the best.

- We identified the optimum Deferred Bayesian Network configuration could be achieved through training with the Genetic Algorithm using 10 chromosomes;
- We found that the use of a Genetic Algorithm with a population of 100 chromosomes is optimal for training the Neural Network;
- Through experimental evaluation, we found that the Non-Monotonic Reasoning classifier obtained the highest cleaning rate for the ambiguous false-positive RFID anomalies;
- Put forth a hypothesis derived from our results that the Non-Monotonic Reasoning classifier succeeded as the superior cleaner due to its deterministic nature being able to handle false-positive RFID anomalies effectively.

While in this work we concentrated on false positive RFID anomalies, the presented concept is applicable to any ambiguous data. While the Non-Monotonic Reasoning performed the best and achieved the highest cleaning rate other methods, due to their nature, would be applicable for other anomalies. Specifically, the Bayesian and Neural Networks would be better suited for an application in which the data is missing, such as false negative anomalies. False-Negative readings need such probabilistic nature to counteract the missing observations which have not been recorded. However, the false-positive anomalies have the problem of gaining too much observations which, in turn, makes a deterministic methodology superior in terms of correcting the anomalies where there is enough evidence to correctly discover the right course of action. Our concept would also benefit other applications that rely on the identification and handling of false-positive anomalies such as physical intrusion detection.

With regards to future work, we believe it would be valuable to investigate effectiveness of other classification techniques, such as the Support Vector Machine, or to develop dedicated classification

algorithm tailored specifically for false-positive RFID anomalies. Also, we would like to investigate the effectiveness of the data correction process when enhancing both the feature set definition or the DMRA data storage, and how our system would react to Terabytes of data as opposed to the test cases we experimented with. It would also be interesting to see the effectiveness of hybrid technique of probabilistic and deterministic methodologies, such as using the Neural Network then passing the findings to the Non-Monotonic Reasoning logic engines. We also intend to incorporate advanced data mining algorithms that have been utilised in past literature, such as Frequency/Statistical based methodologies, as we believe it would enhance the classifier results.

Acknowledgment

This research is partly sponsored by ARC (Australian Research Council) grant no DP0557303.

References

- [1] Y. Bai, F. Wang, and P. Liu. Efficiently Filtering RFID Data Streams. In *CleanDB*, 2006.
- [2] D. Billington. Propositional Clausal Defeasible Logic. In *European Conference on Logics in Artificial Intelligence (JELIA)*, pages 34–47, 2008.
- [3] D. Billington, V. Estivill-Castro, R. Hexel, and A. Rock. Non-monotonic Reasoning for Localisation in RoboCup. In C. Sammut, editor, *Proceedings of the 2005 Australasian Conference on Robotics and Automation*, December 2005.
- [4] M. Blumenstein, X. Y. Liu, and B. Verma. An Investigation of the Modified Direction Feature for Cursive Character Recognition. *Pattern Recognition*, 40(2):376 – 388, 2007.

- [5] B. Carburnar, M. K. Ramanathan, M. Koyuturk, C. Hoffmann, and A. Grama. Redundant Reader Elimination in RFID Systems. In *SECON*, 2005.
- [6] D. Cha, M. Blumenstein, H. Zhang, and D.-S. Jeng. A Neural-Genetic Technique for Coastal Engineering: Determining Wave-induced Seabed Liquefaction Depth. In *Engineering Evolutionary Intelligent Systems*, pages 337–351. 2008.
- [7] S. S. Chawathe, V. Krishnamurthy, S. Ramachandran, and S. E. Sarma. Managing RFID Data. In *VLDB*, pages 1189–1195, 2004.
- [8] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1):1–6, 2004.
- [9] P. Darcy, B. Stantic, and A. Sattar. A Fusion of Data Analysis and Non-Monotonic Reasoning to Restore Missed RFID Readings. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2009)*, pages 313–318, 2009.
- [10] P. Darcy, B. Stantic, and A. Sattar. Augmenting a Deferred Bayesian Network with a Genetic Algorithm to Correct Missed RFID Readings. In *Malaysian Joint Conference on Artificial Intelligence (MJCAI 2009)*, pages 106–115, 2009.
- [11] P. Darcy, B. Stantic, and A. Sattar. Applying a Neural Network to Recover Missed RFID Readings. In *Australasian Computer Science Conference (ACSC 2010)*, pages 133–142, 2010.
- [12] W. Embry. Are you ready for RFID? The promise and challenges of implementing Radio Frequency Identification (RFID) systems across the extended supply chain. Technical report, SAS, 2005.

- [13] D. W. Engels. On Ghost Reads in RFID Systems. Technical Report AUTOIDLABS-WP-SWNET-010, Auto-ID Labs, September 2005.
- [14] J. Holland. *Adaption in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [15] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, April 1992.
- [16] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin. Adaptive Cleaning for RFID Data Streams. In *VLDB*, pages 163–174, 2006.
- [17] N. Khoussainova, M. Balazinska, and D. Suciu. Probabilistic RFID Data Management. *UW CSE Technical Report UW-CSE-07-03-01*, March 2007.
- [18] B. Lam and NICTA. SPINdle [online]. NICTA, 2009. Available from: <http://spin.nicta.org.au/spindleOnline/demo.html> [Accessed: 25th October 2009].
- [19] S. Liu, F. Wang, and P. Liu. A Temporal RFID Data Model for Querying Physical Objects. Technical Report TR-88, TimeCenter, 2007.
- [20] W. S. Mcculloch and W. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysic*, 5:115–133, 1943.
- [21] D. Niedermayer. An Introduction to Bayesian Networks and their Contemporary Applications [online]. niedermayer.ca, December 1998. Available from: <http://www.niedermayer.ca/papers/bayesian/index.html> [Accessed: 2nd October 2008].
- [22] V. Raman and J. M. Hellerstein. Potter’s wheel: An interactive data cleaning system. In *VLDB*, pages 381–390, 2001.

- [23] J. Rao, S. Doraiswamy, H. Thakkar, and L. S. Colby. A Deferred Cleansing Method for RFID Data Analytics. In *VLDB*, pages 175–186, 2006.
- [24] M. Raskino, J. Fenn, and A. Lenden. Extracting Value From the Massively Connected World of 2015. Technical Report G00125949, Gartner Research, April 2005.
- [25] A. Rooij, R. Johnson, and L. Jain. *Neural Network Training Using Genetic Algorithms*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1996.
- [26] D. Rumelhart, G. Hinton, and R. Williams. Learning Representations by Back-Propagating Errors. *Nature (London)*, 323:533–536, 1986.
- [27] H. Stockman. Communication by Means of Reflected Power. In *Institute of Radio Engineers (IRE)*, pages 1196–1204, 1948.
- [28] C. Swedberg. Hospital Uses RFID for Surgical Patients. *RFID Journal*, July 2005. Available from: <<http://www.rfidjournal.com/article/articleview/1714/1/1/>> [Accessed: 5th June 2008].
- [29] F. Wang and P. Liu. Temporal Management of RFID Data. In *VLDB*, pages 1128–1139, 2005.
- [30] R. W. Williams and K. Herrup. The Control of Neuron Number. *Annual Review of Neuroscience*, 11(1):423–453, 1988.

EPC	Reader ID	Timestamp
T1	R3	2009-11-05 16:20:14.032
T1	R4	2009-11-05 16:21:56.012
T1	R2	2009-11-05 16:22:23.005
T1	R4	2009-11-05 16:23:41.043
T1	R1	2009-11-05 16:24:01.009
T1	R1	2009-11-05 16:25:23.004

Table 1: An example of the type of ambiguous scenario our intelligent false-positive removal technique will be targeted to.

	Keep Value	
	True	False
b.loc \leftrightarrow x.loc	22%	88%
c.loc \leftrightarrow x.loc	50%	50%
b.time \Rightarrow x.time	64%	36%
c.time \Rightarrow x.time	42%	58%
b.loc \Rightarrow x.loc	73%	27%
c.loc \Rightarrow x.loc	21%	79%
a.loc \leftrightarrow x.loc	5%	95%
d.loc \leftrightarrow x.loc	93%	7%
b.time \leftrightarrow x.time	33%	67%
c.time \leftrightarrow x.time	46%	54%

Table 2: An example of how the Bayesian Network appears in tabular form.

Rule No.	Rule	Conclusion
1	$c.time \leftrightarrow x.time \wedge \sim c.loc == x.loc$	keep_val
2	$b.time \leftrightarrow x.time \wedge \sim b.loc == x.loc$	keep_val
3	$\sim b.loc \leftrightarrow x.loc \wedge \sim c.loc \leftrightarrow x.loc \wedge \sim a.loc \leftrightarrow x.loc \wedge \sim d.loc \leftrightarrow x.loc$	\sim keep_val
4	$\sim b.loc \leftrightarrow x.loc \wedge \sim c.loc \leftrightarrow x.loc$	\sim keep_val
5	$a.loc \leftrightarrow x.loc \wedge d.loc \leftrightarrow x.loc$	keep_val
6	$b.loc \leftrightarrow x.loc \wedge c.loc \leftrightarrow x.loc$	keep_val
7	$b.loc \leftrightarrow x.loc \wedge c.loc \leftrightarrow x.loc \wedge \sim b.time == x.time \wedge \sim c.time == x.time$	keep_val
8	$c.time == x.time$	\sim keep_val
9	$b.time == x.time$	\sim keep_val
10	$b.time == x.time \wedge c.time == x.time$	\sim keep_val
11	$b.loc \leftrightarrow x.loc \wedge c.loc \leftrightarrow x.loc \wedge \sim b.time == x.time \wedge \sim c.time == x.time$	keep_val
12	$b.loc == x.loc \wedge c.loc == x.loc \wedge b.time == x.time \wedge c.time == x.time$	\sim keep_val

Table 3: The table containing all the rules and respective conclusions utilised in the Non-Monotonic Reasoning Engines.

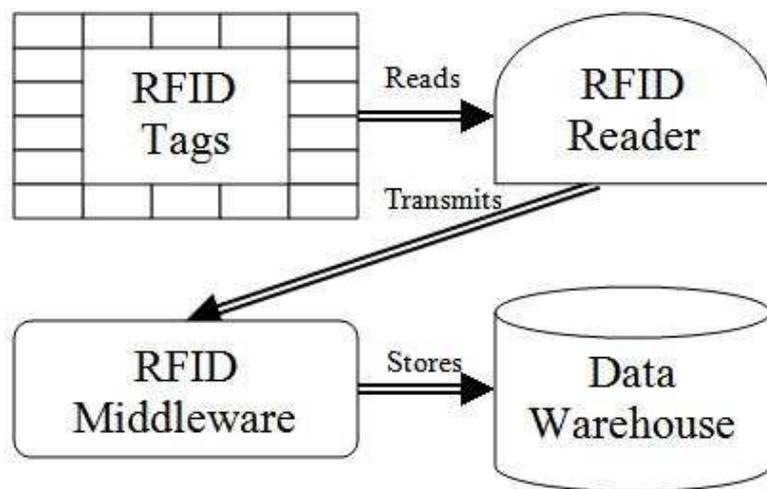


Figure 1: A graphical representation depicting how the data is transmitted through the hardware of RFID-enabled systems.

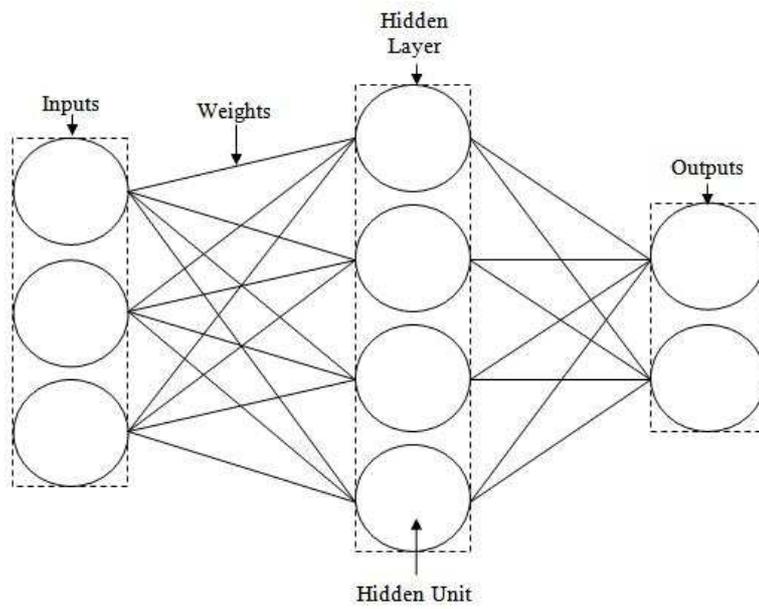


Figure 2: A high-level interpretation of how a Neural Network is designed with its three main layers: the input, hidden and output layers.

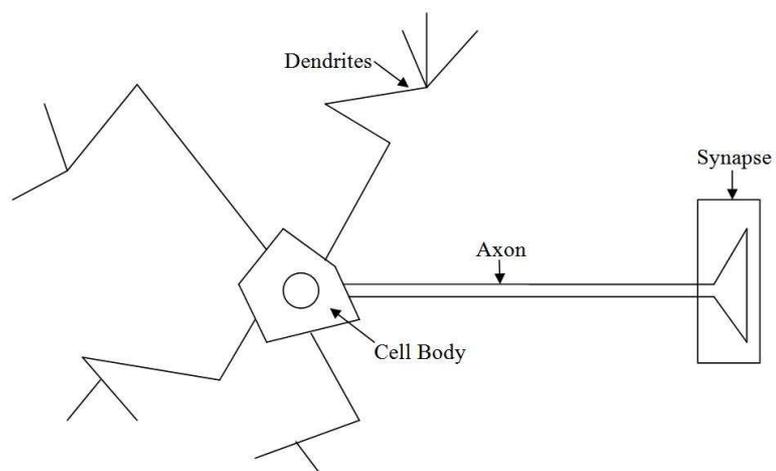


Figure 3: A graphical representation of the main components of a biological neuron within the brain of a living organism.

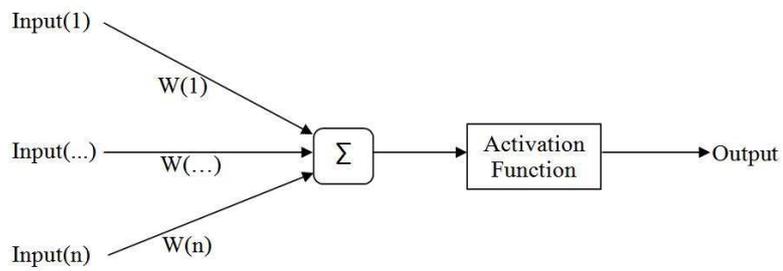


Figure 4: A graphical representation of a digital neuron which has been designed specifically to emulate the processes involved within the biological neuron.

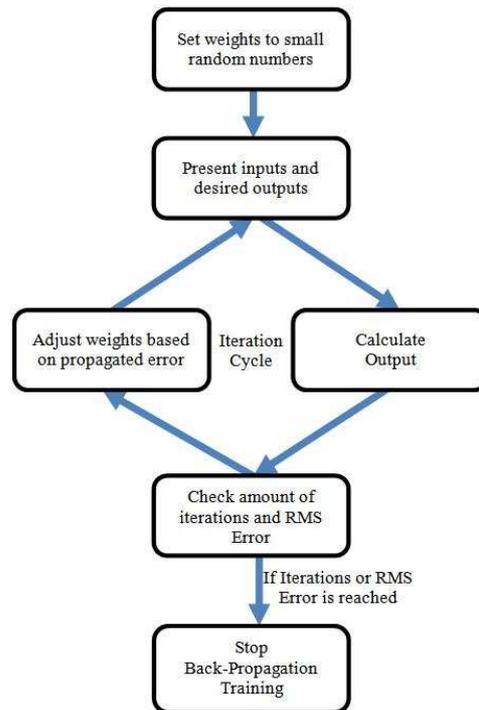


Figure 5: The flow diagram of the processes involved when training a Neural Network utilising the Back-Propagation Algorithm.

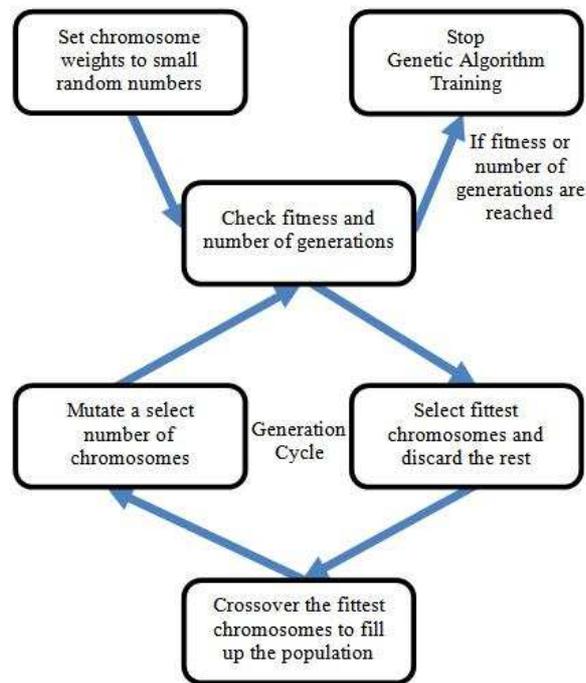


Figure 6: The flow diagram of the processes involved when training a Neural Network utilising a Genetic Algorithm.

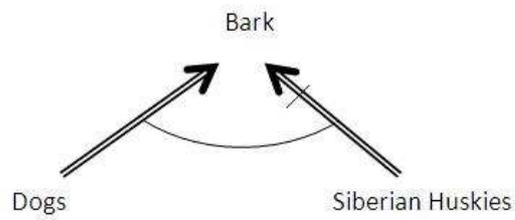


Figure 7: The Logic Map that houses the clausal defeasible rules used when deciding if the Siberian Huskies will bark due to it being a dog or not.

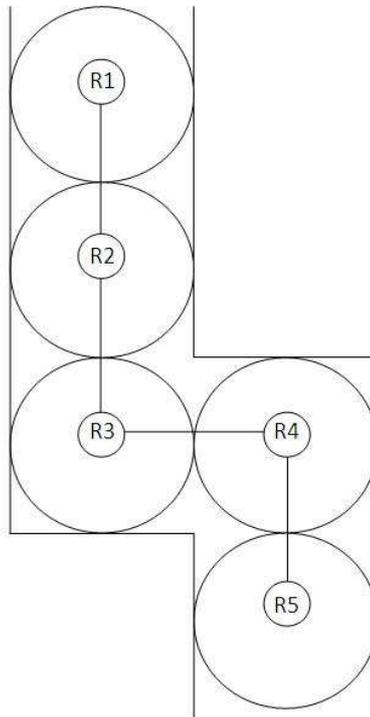


Figure 8: A birds-eye view of an example corridor using RFID to track items designed to illustrate the motivation of our methodology.

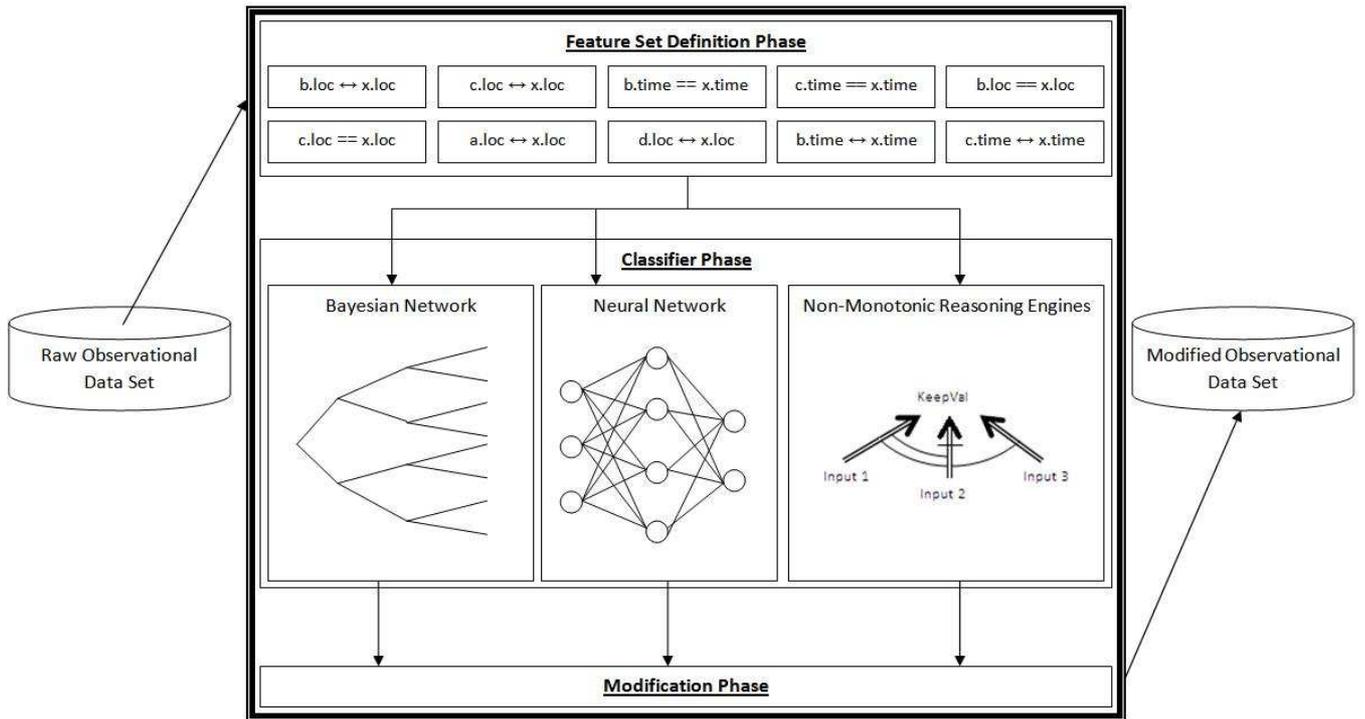


Figure 9: A high-level representation of the different phases the program uses when cleaning the faulty RFID data.

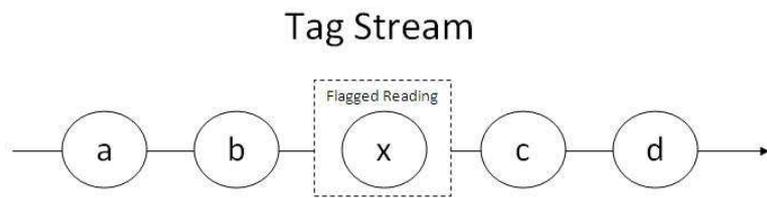


Figure 10: A graphical representation of a tag stream with the observations that are to be examined highlighted as a , b , x , c and d .

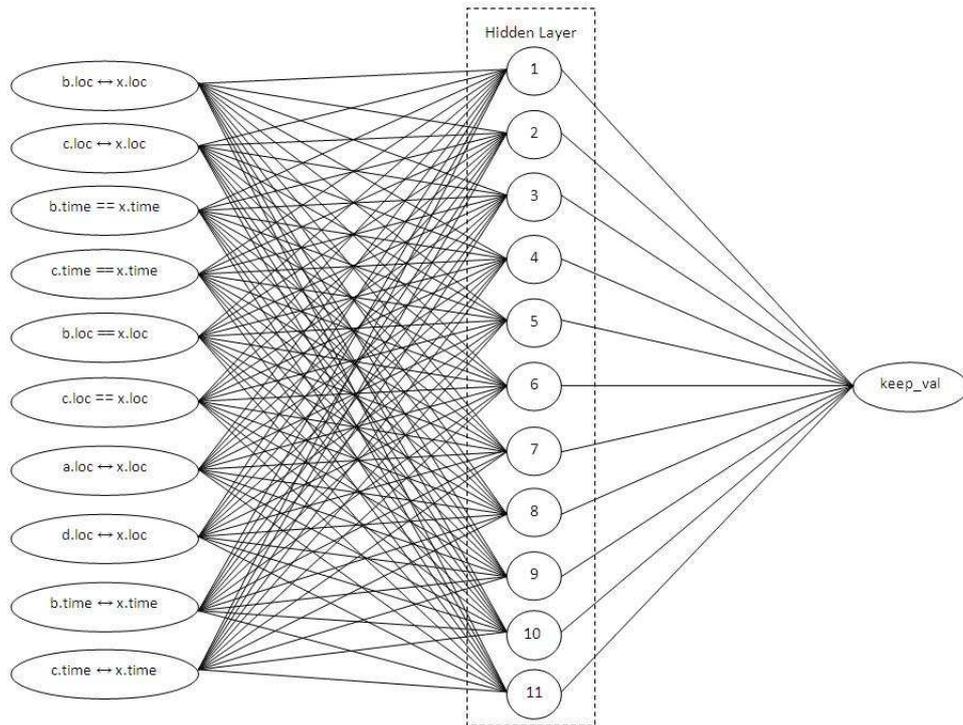


Figure 11: A graphical representation of the Neural Network used in the experimentation of this research.

Bayesian Network Training Results

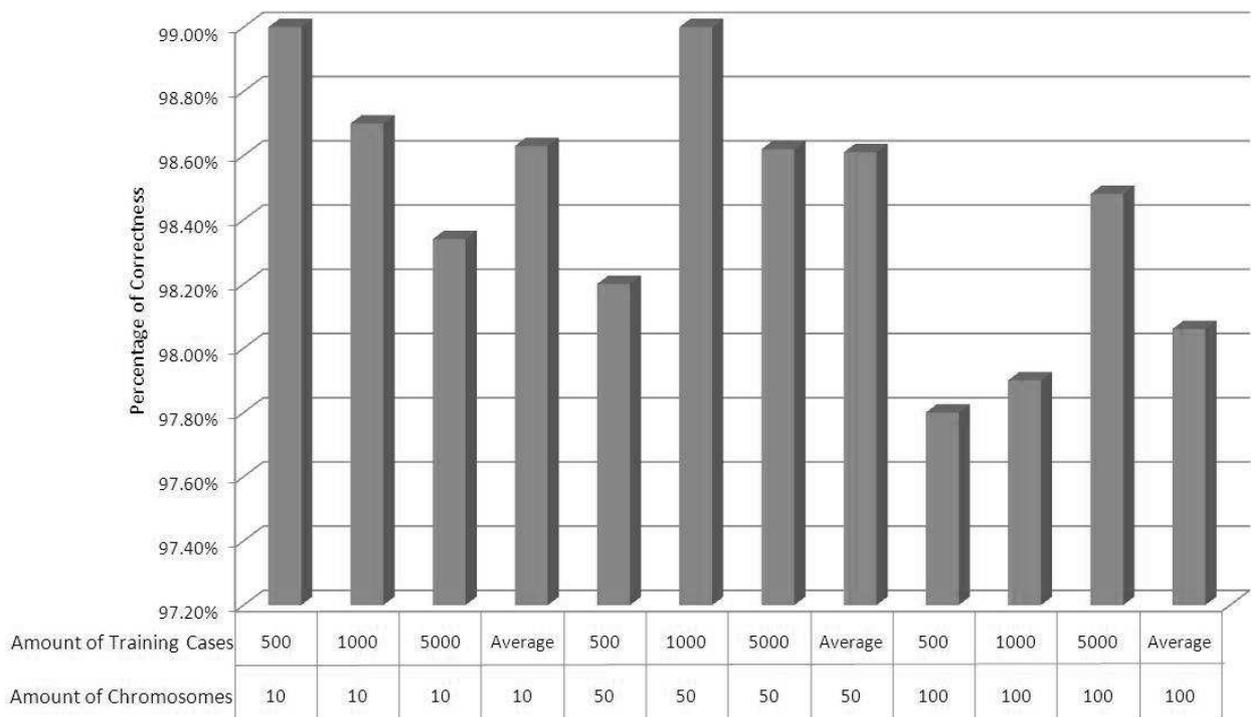


Figure 12: The experimental results of the first experiment designed to find the highest performing configuration of the Bayesian Network.

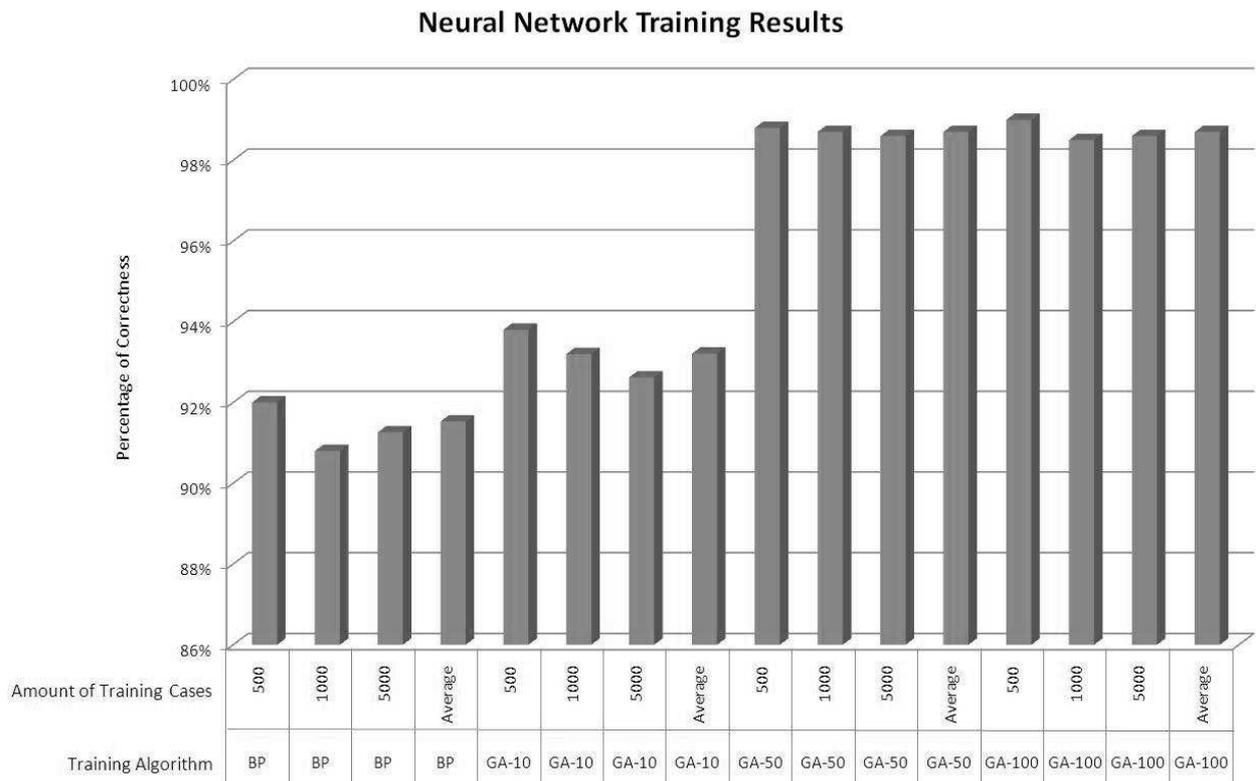


Figure 13: The experimental results of the second experiment designed to find the highest performing configuration of the Neural Network.

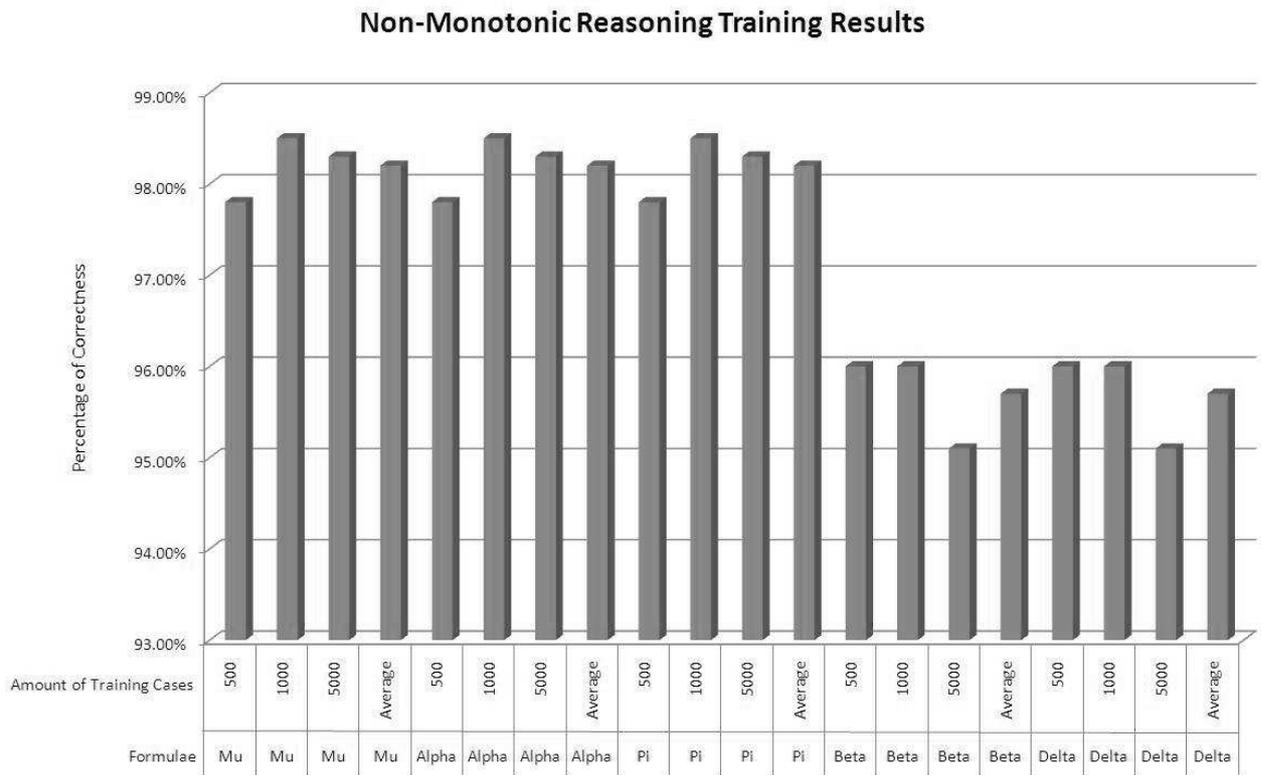


Figure 14: The experimental results of the third experiment designed to find the highest performing formula from the Non-Monotonic Reasoning Engine.

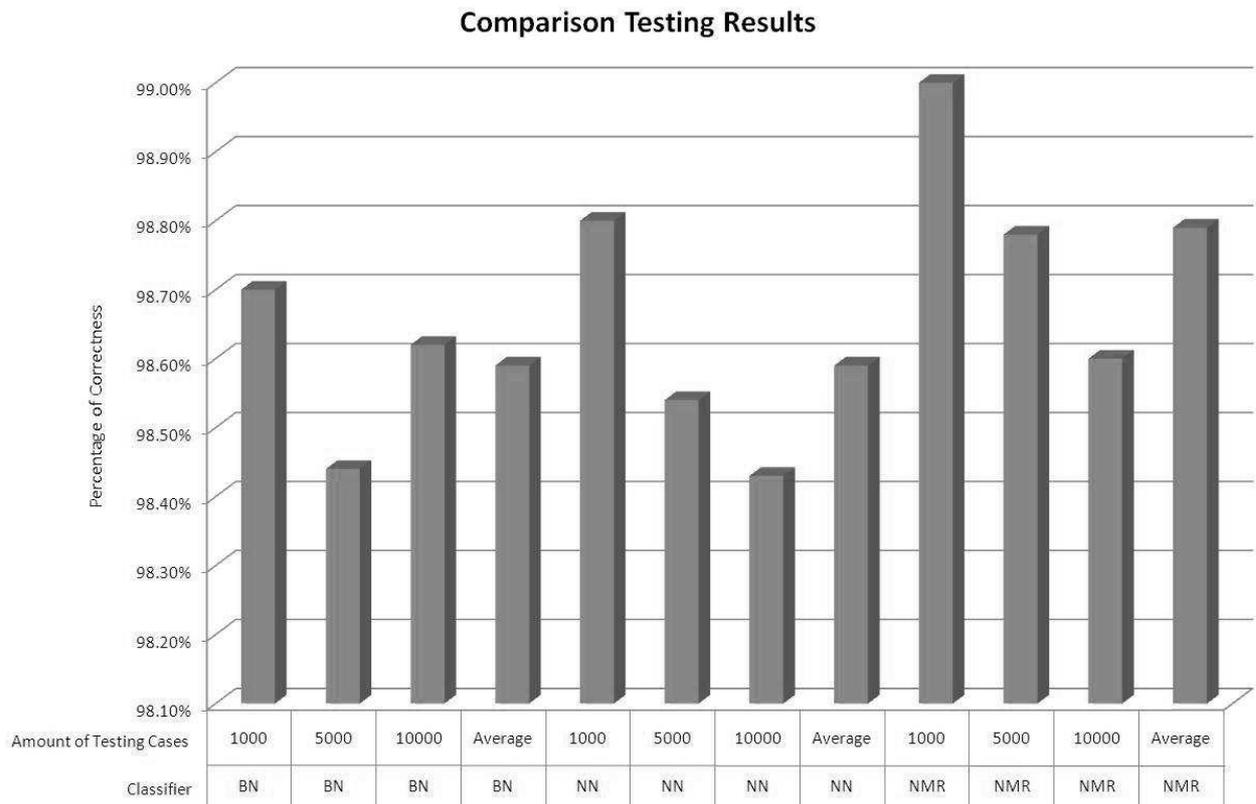


Figure 15: The experimental results of the fourth experiment designed to find the highest performing classifier when faced with a large amount of false-positive anomalies.