2013

Online Deception Detection Using BDI Agents

Richard Alan Merritts
Nova Southeastern University, ramerritts@yahoo.com

Online Deception Detection Using BDI Agents

by

Richard A. Merritts

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Information Systems

Graduate School of Computer and Information Sciences
Nova Southeastern University

2013

We hereby certify that this dissertation, submitted by Richard Merritts, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

James D. Cannady, Ph.D.
Chairperson of Dissertation Committee

Date

Sumitra Mukherjee, Ph.D.
Dissertation Committee Member

Date

George E. Thurmond, Ph.D.
Dissertation Committee Member

Date

Approved:

Eric S. Ackerman, Ph.D.
Dean, Graduate School of Computer and Information Sciences

Date

Graduate School of Computer and Information Sciences
Nova Southeastern University

2013

An Abstract of a Dissertation Submitted to Nova Southeastern University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Online Deception Detection Using BDI Agents

by
Richard A. Merritts
September 2013

This research has two facets within separate research areas. The research area of Belief, Desire and Intention (BDI) agent capability development was extended. Deception detection research has been advanced with the development of automation using BDI agents. BDI agents performed tasks automatically and autonomously. This study used these characteristics to automate deception detection with limited intervention of human users. This was a useful research area resulting in a capability general enough to have practical application by private individuals, investigators, organizations and others. The need for this research is grounded in the fact that humans are not very effective at detecting deception whether in written or spoken form. This research extends the deception detection capability research in that typical deception detection tools are labor intensive and require extraction of the text in question following ingestion into a deception detection tool. A neural network capability module was incorporated to lend the resulting prototype Machine Learning attributes.

The prototype developed as a result of this research was able to classify online data as either “deceptive” or “not deceptive” with 85% accuracy. The false discovery rate for “deceptive” online data entries was 20% while the false discovery rate for “not deceptive” was 10%. The system showed stability during test runs. No computer crashes or other anomalous system behavior were observed during the testing phase. The prototype successfully interacted with an online data communications server database and processed data using Neural Network input vector generation algorithms within seconds.

Acknowledgements

I would like to express my gratitude to Dr. James Cannady for providing excellent guidance as my Dissertation Chair. I would also like to extend my gratitude to Dr. Sumitra Mukherjee and Dr. George Thurmond, II for their support and encouragement as members of the Dissertation Committee. All worked as a team to influence my activities and guide the research process in the right direction from start to finish.

I would also like to thank my family for their support and especially Karen. The dissertation process can be very stressful and strong family encouragement and enthusiasm to complete the program are key ingredients for success. Special thanks to Aynsley and Richard, Jr. for making me laugh.

Table of Contents

List of Tables

Tables

List of Figures

Figures

Chapter 1

Introduction

Background

The Internet is often used by people involved in criminal, terror, fraud, harassment and other malicious conduct (Boongoen, & Shen, 2009). Deception in online data communications is commonly used by such persons as a precursor to the commission of these acts. The word deception can be described as information that falsely represents a fact and is intended to mislead the person to whom it is presented. Because online data has the persistence quality, deceptive information contained therein has the ability to deceive many persons over its lifetime. This quality remains until the deception is discovered and eliminated. If unchecked, online data communications deception persistence allows repeated deception over a broad scope of people and entities that encounter it. Verbal deception is only effective as long as the source of the untruth continues to tell the lies. If others continue the deception it becomes rumor and loses its effect. Deceptive data communications can influence individual's intent on relying on Internet based information for decision making to make poor decisions based on faulty information or become victims of Internet crimes (Jensen, Burgoon, & Nunamaker, 2010).

Research conducted in an effort to protect against deception in online data communications is gaining in importance as people and organizations fall victim to Internet based malicious activities more and more. The ability to detect deception rapidly may serve to slow or disallow the dissemination of deceptive information. Rapidly refers

to the amount of time that an individual needs to determine whether a deception has occurred using automated tools. The U.S. Government has a keen interest in identifying deception in online data communications as part of its Information Operations (IO) initiative. IO is defined by the U.S. Government below:

> Information operations (IO) are described as the integrated employment of electronic warfare (EW), computer network operations (CNO), psychological operations (PSYOP), military deception (MILDEC), and operations security (OPSEC), in concert with specified supporting and related capabilities, to influence, disrupt, corrupt, or usurp adversarial human and automated decision making while protecting our own (U.S. Department of Defense, Joint Chiefs of Staff, 2006, p. ix).

Online data communications deception and its discovery falls within the CNO subcategory.

Problem Statement

Current developments in deception detection in data communications environments are ineffective due to their time intensive data extraction processes and inefficient approach to processing data which is not suitable for deception detection in real time data communications environments. Automated linguistic and other indicators of deception have not been developed and deployed in online communications environments (Zhou & Zhang, 2008).

Dissertation Goal

The research goal is to create capabilities which autonomously and automatically detect deception in online data communications systems. The capabilities developed will

detect online data communications deception to realize this goal. To measure accuracy, evidence of a deception will be uncovered in text entered by the user.

Relevance and Significance

Deception in online data communications will continue to increase if automated deception detection systems are not developed to combat it. All people and organizations are susceptible to online data communications deception resulting in criminal activity and more will be victimized as the Internet continues to grow. For example, nearly one billion pounds ($1.58 billion US) have been lost due to theft by criminals using online deception as a means to commit crimes in the United Kingdom (Boongoen & Shen, 2009). Exacerbating this problem is the fact that some online data communications are completely harmless white lies, while others are the precursors to serious crime. This is a difficult problem to solve. Finding a solution is vital. Many will benefit from research efforts undertaken with the goal of deception detection capabilities in online data communications (Zhou & Zhang, 2008). Some deception detection research efforts have met with success, but they lacked automation and autonomous features (Jensen, Burgoon, & Nunamaker, 2010). Non-automated deception detection research is relevant to this effort because the techniques used are valid and effective and have exposed an automation gap in the research. This gap has been addressed using BDI agents for automation in this study. As such this study has extended research and advanced deception detection in online data communications.

Barriers and Issues

Detection of deception capabilities are difficult to create due to the rapid evolution in technologies and media which support online data communications methods.

Also, unaided by computer automation, humans could not analyze and synthesize the amount of data necessary to be effective in deception detection. According to Zhou & Zhang (2008) real-time and near real-time automated deception detection solutions are yet to be developed.

One other barrier to the research that was conducted was the testing of BDI agents. This research used a dynamic BDI agent testing process in a real-world setting. The traditional static model BDI agent testing method was not used. In contrast static model BDI agent testing is conducted out of band away from the operational environment affected by the test. The dynamic testing methods for BDI agent development caused complexity in determining whether the operational environment could have adversely affected the agents. However, the risks were identified and when problems arose mitigation techniques were employed (Balke, De Vos, Padget, & Traskas, 2011). The dynamic BDI agent evaluation model was selected for agility and flexibility and was the correct selection for this deception detection study.

The above barriers intensified the difficulty of the research and added complexity to the development effort. This has been brought to light in the literature review portion of this study. Within the literature review other deception detection in online data communications such as email and agent development using modeling and BDI agent architecture solution research efforts are examined and discussed (Fette, Sadeh, & Tomasic, 2007; Meneguzzi & Luck, 2009; Thangarajah et al., 2008).

Limitations

This research was restricted to deception detection in data communications in an online environment. The prototype developed relied on BDI agents to automate tasks in

the deception detection process. Although BDI agents were not able to learn or dynamically change, they were employed successfully for automation despite their limitations. The BDI agents were reprogrammed to adapt to changes during system development to overcome their shortcomings.

Delimitations

BDI agent capabilities were identified for this research for the purpose of delimitation. The BDI agent abilities were used to automate deception detection tasks in online data communications. The focus of the study was strictly limited to online data communications although the adaptability to many others is possible due to BDI agent flexibility, modularity and portability features.

Definition of Terms

1. BDI agent – A software program that is capable of autonomous behavior and decision making using a set of beliefs, desires and intentions.
2. Online data communications server – An application server consisting of a web server front to a database server allowing users to post messages to the database for other users to view and respond to.
3. Deception – False representation of fact.
4. Autonomous – Ability to carry out tasks independently without human interaction.
5. Agent plan library – The executable code that a BDI agent relies on to interact with its environment.

Summary

The problem statement, research goals and approach were discussed in this chapter. A major problem in the deception detection in online data communications problem space has been identified and addressed by this research effort. The problem of deception detection in online data communications impacts almost all people and organizations worldwide. The problem continues to grow with the proliferation of the growth of the Internet and reliance on Internet information to make decisions.

Chapter 2

Review of the Literature

Phua, Lee, Smith-Miles, & Gayler (2007) examined online credit fraud in their research. The study involved online credit application deception detection. The study pertains to this body of research because data was extracted from a backend database to an online system lending it similarity in the data container context. Certain database fields were analyzed for indications of deception after credit seekers submitted applications. A Communal Analysis Suspicion Scoring (CASS) algorithm was used to show potential indications of deception within the database entries submitted. Criminals typically use three methods to commit fraud in online credit applications:

- Identity theft
- False identity creation
- A combination of both

The CASSS algorithm is not adaptive enough to change rapidly with the dynamic capabilities of criminals intent on committing credit fraud, exposing a weakness of the study. This weakness could be overcome using Neural Networks (NN) and autonomous and intelligent agent usage capabilities in addition to CASS. Increased automation and flexible capabilities included in approaches can be achieved using BDI agents to advance the technology and NNs can be used for adaptability to bridge this adaptability gap.

The parallel area of Phishing detection research was conducted by Fette, Sadeh, & Tomasic (2007). In their research a method of detecting Phishing data contained in email was developed. Detecting Phishing email research is differentiated from other deception

detection techniques. Phishing emails must possess the ability to appear genuine to fool the recipient. Subtleness in appearance is a key feature of Phishing emails. In the research, invalid URLs were sought. False URLs were contained in the Phishing email intended for users to select them and be redirected to a potentially harmful Web site. Bogus Domain Name System (DNS) entries were found to be common in Phishing emails and the researchers addressed this as well. A weakness of the research was the lack of employment of linguistic cues to deception which is used in many other deception detection studies. The researchers deemed linguistic cue based deception detection in phishing email research inappropriate since actual deception may not be contained in the email. The Phishing email detection research is pertinent to this study because Internet based information was being studied. Embedded malicious URLs were determined to be the major thrust of Phishing email detection research. Test data used for the study was known Phishing email. The researchers developed a prototype and integrated it onto an email server. Machine learning techniques were used to classify the embedded URLs as malicious or not. The Phishing research approach is pertinent to this study because a prototype was integrated into an online data communications server.

According to Zhou & Zhang (2008) criminals choose email as the medium when committing fraud using deception in the majority of cases. Linguistic cues to deception detection were developed to determine if lies were embedded in email. There are three considerations that aid deceptive individuals in attempts to deceive using online data communications:

1. Unavailability of non verbal cues.
2. Absence of a social interaction.

3. Expiration of time since the deceiver has more time in formulating a deceptive response in online data communications vice during a face to face exchange.

The researchers reveal that research is very uncommon regarding automation of detection of deception in domains like data communications online. The researchers state that general, manual deception detection research capability development research is much more common. According to the researchers deception detection research capabilities combined with intelligent agent research may assist in finding viable automated solutions. The study listed characteristics of linguistic cues to deception within text. Paralinguistic indicators were also identified. These are keyboard activities, including back spacing and deleting actions which are much more common for those being deceptive in online data communications as they craft their deceptive communications. The study identifies online deception indicators as verbal and non verbal. Text content is identified as a verbal classification. The researchers failed to capture a link between verbal and non verbal deception indicators because activities such as keyboard captures can yield paralinguistic cues to deception. Failure to identify the link between linguistic indicators and paralinguistic indicators is a major weakness of the study. Further, intelligent agents could be used to advance the capability by capturing paralinguistic cues and using Machine Learning to link these cues to linguistic cues. BDI agents could automate much of the paralinguistic indicator capturing activities.

Deceivers using online data communications tend to have more of a sense of security than face to face deceivers due to the physical and geographic separation between them and those they are deceiving according to Zhou and Zhang (2008). The researchers explain that there is less emotional stress on the deceivers due to the

knowledge of this separation. A research gap has been identified by the study in that efforts to combat online deception are hindered by rapidly changing technology behind the data communications. One positive factor is the commonality of using text in online data communications. But the way in which the text is drafted and transmitted differs. For example instant message traffic must be generated quickly and rapidly communicated to be effective. Weblog users on the other hand can take their time to formulate entries. The study illustrated the characteristics of deceptive text found in online data communications text. This finding is pertinent to the research described within this report since the characteristics of deceptive text may be universal across media types.

Boongoen & Shen (2009) studied identity theft deception detection. In the United Kingdom alone over one billion pounds ($1.58 billion US) were stolen using online deception resulting from or combined with identity theft. The study also examined identity theft used by terror groups. The researchers developed a connected path algorithm based prototype to map social networks. Anomalies in the social networks are identified as potential deceptive entities. Cardinality of neighbors is typically used to determine anomalies within social networks so the connected path was a non typical research approach. The connected path algorithm employs dynamic link analysis and determines if the link anomalies are supported by probability calculations. The researchers assert that online deception is the starting point for most identity theft crimes. The relevance of this research to the researched described in this report is that it is vital to detect deception early in the identity theft lifecycle in order to prevent it. Terrorists use the Internet to disseminate plans, propaganda and boasting of their exploits. Investigators could use deception detection tools to determine validity of claims, terror plans and true

identities if deception detection tools are available. These types of tools could be added to investigative and anti terror tools to protect national interests, people and property. Since these tools are not fully developed, this is a major research gap.

An interesting parallel research area is deception detection in online dating forums as studied by Toma & Hancock (2010). The research into online dating deception detection examined user profiles. The user profile data contains descriptions of the web site hopeful who is looking for a compatible dating match. Intentional deceptions in the self representation portion of the profile were found with an underlying motivation being to appear more appealing and desirable. There were differences in male profile deceptions and female profile deceptions. Marital and relationship status as well as height deceptions were found to be high in male profiles. Deceptive information about weight and uncharacteristic photos were the norm for female patrons. The researches relied on deceptive characteristics of online dating which was imported from another study. Building on previous research is an important consideration for all research efforts. Cognitive and emotional subtypes of linguistic cues to deception in online data communications were uncovered. Guilt and negative emotional states of the deceivers were manifested in the writings of the deceivers and resulted in text based indicators of deception according to the researchers. The conclusions reached in the research bolster the significance of online deception detection studies. This research furthered deception detection capabilities by developing cognitive indicators of deception. The researchers concluded that less expressiveness and more action words are used while formulating lies in text entered in online dating profiles. The researchers found that online dating profile deceptive text contains more words on average than truthful entries. These observations

were taken into consideration during the research conducted in this report. For the online dating profile deception detection effort, manual text extraction was used. The data was placed in text files which were loaded into a deception detection system. Emotional linguistic cues to deception were the best indicators of untruthful profile renderings within online dating forums.

Jensen, Burgoon, & Nunamaker (2010) researched a prototype known as the Behavioral Analysis Prototype (BAP). BAP is a system capable of synthesizing interview transcripts. The interviews are law enforcement investigation related. Video and audio from the interviews is studied. BAP is used as a helper tool for law enforcement to increase the ability of detecting deceptions which were contained in transcripts combined with analysis of the body movements of the subject of the video. BAP is a Decision Support System (DSS) with a combined video analysis and text analysis of transcripts. BAP increased deception detection during implementation and was an effective DSS. BAP is not automated and a great deal of labor goes into transcribing the audio and analyzing the video from interviews.

BAP was not designed to autonomously make decisions about deception; instead it is a tool to help assist law enforcement personnel decide if they are being lied to. The transcription creation process is time consuming, poses a major weakness to the study and exposes a research gap as BAP would not be suitable for deception detection in online data communications. In an online data communications environment Kinesics data or body language data would not be obtainable. BAP is heavily reliant on kinesic data embedded in the video portion of the interview. However, the analysis of the deceptive text contained in the transcripts of interviews is pertinent to the research

described in this report. Characteristics of BAP text were relied upon for this research effort. BAP had no autonomous features. BAP relied on humans to manipulate the data and make the final deception detection determination. This highlights the need for more autonomous behavior in deception detection system development.

Marnazato, Pereira, Neubert, & Pereira do Lago (2010) studied fraud detection in commerce systems hosted online which contain reputation rating capabilities. The TodoOfera system similar to eBay but servicing South America was the focus of the study. The researchers examined user attributes to determine which ones might be likely to conduct fraudulent transactions. The researchers began by gathering data on users who had been previously been banned from using TodoOfera. The researchers then binned them into categories based on the reasons behind the blockage. Machine usage information was gathered to determine if various users were sharing machines. A logistic regression algorithm was developed to determine the probability that users would rate products and services falsely. Data samples were derived from user characteristics, registration data and time sequence of ratings after transaction commission. The researchers tied the findings of their logistic regression algorithm to user fraud activity based on rating transactions. The researchers used what is known as warrant data or data contained in user registrations. No text based deception detection was attempted, exposing a major flaw in the study. Text based deception detection could be beneficial for ratings of transactions that allow textual inputs.

This research prototype system development effort employed BDI agents for automation in deception detection in online data communications content. The next few reviews summarize and analyze BDI agent research and their associated methods,

processes and characteristics. This gives the reader insight and understanding of the BDI automation capabilities and possibilities during design and implementation.

Research by Thangarajah, Harland, Morley, & York-Smith (2008) extended a BDI agent architecture by developing an individualized BDI agent framework employing only modeling language. The researchers used the CAN BDI language. Pseudo code was developed during the research which extended the BDI agent architecture. The CAN code was used to test the logic of the architecture extension, but CAN is not executable. The researchers planned to use SPARK agent development software to import the CAN pseudo code in an effort to test the algorithms during future research. The researchers used industry standard development methods employing a building block approach, first working out details in pseudo code to test logic, then moving on to write an executable code base. The effectiveness of the developmental research method was demonstrated in the study by showing how continuity can be built in to the design process.

Taghezout & Adla (2008) studied adaptive user interfaces with BDI agents embedded for industrial usage. Internet information was successfully gathered by the BDI agents. The BDI agents synthesized the Internet gathered information for presentation into an adaptive user interface. Complex data set analysis assistance for operators was employed using the BDI agents from within an industrial system. The BDI agent role was to capture critical information for the users and eliminate confusion from data overload based on the users' continually changing needs. Users were not exposed to vast quantities of data as the BDI agents were able to effectively make decisions on the importance of the data presented to the users. The study showed the suitability of BDI agents for

potential usage in eliminating confusing data in the area of deception detection. The study gave an excellent explanation of why BDI agents were chosen for the user interface adaptability tasks. BDI agent autonomous decision making features were the primary driver for the decision to employ them, combined with agent teamwork capabilities. The teamwork capability and the autonomous decision making capability of BDI agents were pertinent in the decision to employ them in the prototype development effort described in this report.

Frank (2008) successfully extended the BDI agent atomicity property in his research effort. The Java "extends" statement was employed to add to the atomicity BDI agent property. The researcher named the capability distributed relaxed atomicity as a distributed attribute was included. Agents continued to operate within a distributed cell data communication environment containing a dynamic IP address naming scheme. If the distributed relaxed atomicity property was not present in the BDI agent design, the agents would die once their IP address was not determinable. The relaxed distributed atomicity extension allowed the agents to wait for dynamic IP address assignment in the operational environment. The norm for BDI agent atomicity is for them to cease to operate and abort if beliefs about their environment are not determinable. Using the distributed relaxed atomicity property the research showed how agents could be allowed to wait and anticipate environmental state changes. Since the BDI agents were operating in a mobile phone environment, the capability was essential due to the dynamic nature of the network addressing within and among the various cells. BDI agents facilitated execution of programs on mobile devices as they physically moved within the network.

Meneguzzi & Luck (2009) extended the BDI agent architecture by developing norm based behavior extensions to java based BDI agents. Agents have an extended decision making ability using norm based behavior attributes contained within the agent plan library which is the intentions portion of the BDI agent structure. These intensions contain executable code that the agent can select if it has a plan to address the intended action. The norm based extension restricts the BDI agent's behavior by adding restrictions to the execution of the executables in the plan library unless certain conditions are met. A developmental research methodology was employed during this research. Developmental research is very common in BDI agent research. BDI agent development can be conducted very rapidly allowing for the developmental research approach to be used. The study concluded that BDI agents could successfully use norm based behavior to further refine their intentions. Development was conducted using AgentSpeak BDI agent IDE. The deception detection research contained in this document evaluated the AgentSpeak IDE, but ultimately chose AOS JACK Agent Development software due to design feature considerations. Both AgentSpeak and JACK support developmental research efforts and rapid BDI agent capability development.

Thimm & Krumpelmann (2009) aligned BDI agent beliefs with their intentions by modifying the BDI architecture during their research effort. BDI agent beliefs are the ideas that the agent holds about its environment, in contrast the BDI agent intentions contain actionable executables in a plan library. These intentions or executable plans can facilitate the carrying out of goals or desires. The BDI agent belief system was modified to contain information about what is contained in the plan library rather than just information about the agent environment. The research altered the basic BDI model.

The research goal was to give the agent know-how. That is the ability to have reasoning about how to accomplish goals based on the contents of the plan library. Therefore procedural knowledge had to be incorporated into the agent belief system. This concept is a departure from traditional BDI agent architecture where the beliefs are restricted to environmental constraints and conditions. The current context is an integral part of the BDI agent belief system. The BDI agents' potential achievements are incorporated into the goals. BDI intentions are linked to the desires formulating sub goals which can be achieved through plan library actions in response to events and environment changes. Know-how links the beliefs about the environment with beliefs about actions to achieve sub goals or intentions. Know how incorporated into BDI agent beliefs allows greater autonomy when agents interact with their environment. The underlying concept is that the agent is able to evaluate the effectiveness of whether a plan will be effective or achievable. This capability gives the BDI agent awareness about the contents in the plan library. BDI agent know-how may be relevant to deception detection capability development for online data communications due to the added flexibility of the belief system. Knowledge of the plan library as well as the environment extends the BDI agent reasoning ability.

Patel & Hexmoor (2009) studied BDI agent usage in gaming in the creation of Bots. Bot gaming strategy was incorporated into the BDI agent's structure. BDI agents programmed with gaming strategy offer the game players more challenge as the agent based Bots are less predictable than standard Bots. Capturing strategy to be contained in a BDI agent's plan library was a promising potential capability for the research described

in this report. BDI agent strategy formulation capabilities may aid in future agent based deception detection research.

Ma, Chen, Bai, & Huang (2010) used BDI agents for web service testing in their research. BDI agent teams were designed using a class structure to define their team roles and responsibilities. A Test Runner (TR) combined with a Test Coordinator (TC) team member class structure was developed. BDI agent team coordination and complimentary functions were core components of the web service testing agent research. The BDI agents successfully migrated from a web server to client computers. The researchers used the JADE BDI agent development environment. JADE is compliant with the standards of the Foundation of Intelligent Physical Agents (FIPA) and this was the basis of selection by the researchers. The researchers chose BDI agents to conduct web services evaluation functions since they are capable of detecting and responding to web based environmental condition changes. The BDI agents were also chosen because of their flexibility for use while embedded in a web server plus the ability to migrate to the client using the web services. BDI agent research was successful in deciding courses of actions and processing complex and high volume web service related evaluation tasks. The research is relevant to the research described in this report since it illustrated the autonomous nature of BDI agents in various environments, especially web based environments, which are representative of online data communications environments. BDI agent teamwork and synchronization of activities was also very import to the research conducted as described herein.

Miles & Tashakkori (2010) show how using a Finite State Machine (FSM) implementation based on software based intelligent agents gives the games stability and

testability, while allowing simulated characters to display behavior as actors in a virtual environment. However, the researchers assert that the FSM approach may be too predictable for human game players. This is due to the migration toward games where players do not necessarily interact with agent based characters alone, but rather interact with artificial game players as well as other humans in the game environment. Interacting with other humans offers a dynamic game environment and a degree of unpredictability that the FSM based agent player cannot afford them. Genetic algorithms were developed and tested as an alternative to the predictable FSM approach. An environment similar to the SIMS virtual environment was developed to mimic an operational environment.

Volunteers observed the behavior of a simulated student operating within the system. Agents responded to their environment using needs which varied in importance from 1-20. The system gauged happiness of the agent based on its abilities to meet its needs. Needs were translated by an agent logic engine into states. An agent that needed food was transformed into a hungry agent state. To decide an agent course of action for a given state three methods were developed, random selection, FSM and Genetic Algorithm. To test the effectiveness of the system, surveys were administered to the human observers. The Genetic Algorithm based approach was found to be the most believable of the simulated student in the virtual environment. However, the FSM came in second followed by the random approach. The survey respondents overwhelming felt that the FSM approach was very predictable. This was due to the consistency of behavior of the computer controlled character using the FSM approach. Additionally the results for preference of approach was broken down by frequency of game playing. Participants who play video games every day were found to prefer the Genetic Algorithm approach. As the

research points out, intelligent agent software based FSMs are common, stable approaches to provide predictable, reliable approaches for execution of programs. However, a Genetic Algorithm approach may provide equal stability with an unpredictability aspect that the FSM cannot provide. The FSM's stability is certainly a strength when predictability is not a problem. This may be a positive approach for an Information Assurance application like deception detection in online data communications where stability, testability and predictability of agent activity are a requirement.

Telang, Meneguzzi & Singh (2013) expand the concept of Hierarchical Task Networks (HTN) for BDI agent task commitment. This is a departure from the FSM approaches used for task commitment for BDI agent based systems. These FSM approaches are typically employed to facilitate autonomous behavior. For BDI agent software based FSMs, the researchers consider them to be commitment machines. A commitment machine approach although stable, predictable and testable lacks flexibility in reasoning. However, ease of implementation using the FSM as a commitment machine is facilitated greatly. To illustrate how commitment could be reasoned by intelligent agents using HTN the researchers developed a purchase scenario between a seller and a buyer. The customer has purchase goals and the merchant has acquiring currency in exchange for goods goals. The sequence of payment or receipt of goods may be any order as in many transaction types in the real-world. The HTN allows for the formulation of payment plans and goods delivery options on the fly which are mutually acceptable to both agent types. An FSM lacks this flexibility. The researchers describe the intelligent agent commitment lifecycle which has two active states, conditional and detached. From

the active state the goal can be terminated with a condition of satisfied or violated. Sequencing of actions is handled in the plan library using state transformations. In the HTN context the plan uses refinement of task sets. Compound tasks are broken down until only primitive subtasks remain. This allows for basic method construction which describe how the high level compound tasks will be accomplished using the primitive methods. The result of the research was the development of an HTN framework that can be applied to commercially available HTN planning environments. The main offering is goal and commitment operationalizing. This offers greater flexibility to the FSM approaches typically used in BDI agent implementations where commitment and synchronization of activities is necessary. Synchronization and commitment are achieved using BDI agent software based FSMs. However, more real time options in planning are available using HTN. This is relevant to the deception detection research because more flexibility in planning may be required as adaptability features are added in future implementations. The HTN framework allows for more efficient creation of planning for agents. The framework developed for HTN allows for the protocols to be readable by humans and allows for cost estimation of the protocols for optimization prior to execution.

Sloan, Kelleher & Mac Namee (2011) research Utility-based Control (UBC) for usage in video gaming for Non Player Character (NPC) control. UBC has not been used in controlling NPCs primarily because it is much more resource intensive than FSM or Goal-Oriented Action Planning (GOAP). The reasoning behind researching UBC for NPC usage is that it gives the NPC a degree of unpredictability and adds entertainment value. The researchers consider the popularity of the FSM as the most commonly used

NPC behavior control technique due to its simplicity and conservation of resources. GOAP is described as currently employed, but the most complex and resource intensive NPC behavior control method used. The researchers constructed an environment using NPCs using all three to be able to compare the three approaches objectively. The virtual environment chosen for the NPC behavior control evaluation was a simulated hospital environment. The researchers describe the FSM as having a predefined body of states. These states contain predefined actions that an agent can perform while in that state. Only one state at a time can be maintained. For example in a military game, when an agent starts a patrol state it transitions to an attack state. Once an event occurs such as encountering an enemy, the agent can transition to an attack state. If the agent determines a low health state it can transition to a disengage enemy state or if the enemy killed event is achieved, the agent can transition back to patrol state. Using FSMs requires a definition of every state that an agent based NPC can be in. The sequence of actions that an NPC can do is always fixed and therefore can become very predictable to the human player. FSMs work well and are very efficient when there are fewer states to transition to. FSMs with complex and many states can cause performance problems. During design time, all states must be fully defined because intelligent agents cannot be modified on the fly. This complicates development when complexity of transitions is necessary for game realism. FSMs are environment specific, therefore if a different development environment is being designed, an FSM from a previous build may not be able to be reused. FSMs are criticized by the researchers because they do no support concurrently occurring behaviors.

The researchers describe GOAP as an NPC behavior control capability that allows for runtime decision making about what to do and how to accomplish the necessary tasks. The goal allows the agent to evaluate the situation based on a set of conditions. The GOAP architecture also allows for cost calculations for determining the expense of a potential action. Failed plans are added to the knowledge base to be formulated back into the goal. The failed plans are excluded. Goals are selectable under GOAP and based on a predefined selection mechanism. The goal selector prefers to select the lowest cost plans from the plan library. The weaknesses of GOAP are that specific behaviors are difficult to design since only high level logic programming is available. Another drawback to GOAP is that no action can be initiated while the plans are being evaluated. Only one goal at a time can be initiated.

UBC measures the desirability of agent state within a goal set. Using this method UBC enables mapping of the effect of the utility to the goal set to the action. Those with the highest utility value will be selected. This helps to de-conflict goals, for example safety over speed or when uncertainty exists with weighting for multiple goals. Weighting of goals and success probability for actions are used to select the best agent actions.

For comparison of the FSM, GOAP and UBC, a hospital virtual environment was setup. A mix of NPC using each was designed to represent nurses, doctors and patients. The evaluation involved measuring processing and memory usage of each approach, complexity of the NPC behavior, and ease of implementation and their extensibility. The UBC consumed more processing time than FSM or GOAP; however memory resource usage was about the same between UBC and FSM. UBC used less memory than GOAP.

Behavior known as emergent behavior was displayed by the UBC NPCs. GOAP and FSM NPCs did not display emergent behavior. The complexity of the UBC modeling was far greater than both FSM and GOAP modeling complexity. Scalability for UBC was a problem as well. This research is pertinent to the deception detection research in that it shows how agent software based FSMs are constructed and operated and that they are very stable, despite having limitations in that the agent plans are inflexible at run time.

This section of the literature highlights research efforts employing intelligent agents and Fuzzy Logic in novel and in Information Assurance (IA) applications. BDI agents were used in the prototype developed in the research contained in this report in a deception detection role which is an IA function. It is important to identify other IA research efforts using a similar or parallel approach. This section also exemplifies that developmental research was the most favorable method. The research within this section is relevant because as deception detection in online data communications is developed and proliferates, deceivers may attempt to hide deception within the text making a future adaptive solution essential.

Kuderna, Hoszu, Vacariu & Cret (2009) researched smart house systems by creating a prototype. The approach was eclectic using BDI agents, a NN and sensors embedded in the home for comfort control combined with a smart phone for communications. A brain agent was created to contain the logic needed for the smart house to make decisions. This was combined with a knowledge base agent which contained data necessary for comfort control. The brain agent and the knowledge base agent formed the decision layer of the system. The combining of NN and BDI agents is not characteristic of other research efforts but allows an extension of the decision making

abilities of the agents for adaptability in dynamic sensor reading. This allows for the ability to decide what comfort settings humans would be most inclined to accept. The researches introduced robotics into the system for Human Robot Interaction (HRI) as well as a robotic interface to the system components. Temperature, light, relative humidity and proximity components were managed by a sensor agent. Other agents interacted with system components, provided information mapping and took actions when necessary. This study showed the power of combining intelligent agents with a NN. The system effectively controlled the environment using agents relying on the NN for learning. The human experience with the system was very favorable from a comfort perspective.

Lim, Cheng, Rohatgi & Clark (2009) researched Machine Learning for management of security related policy. The study attempts to cause the system to respond to security policy by overriding irrelevant policy based on situational changes in a dynamic environment. The policy engine is reprogrammed as the Machine Learning algorithms learn to adapt on the fly. Risk factors are considered in the policy revisions as the policy implementations are interpreted and acted upon by the Machine Learning function. Genetic Programming (GP) was used in the developmental research effort due to its ability to search and make decisions from data extracted from large data sets. This GP approach differs from BDI agent approaches in Information Assurance. BDI agents must be programmed with all of the environmental characteristics beforehand to be able to interface with the data containers and execute their activities and respond with their plan library (intentions) code base. The study employed the Multi Object Genetic algorithm framework combined with data mining. Other successful uses of GP were identified in

the study. However, these were for intrusion detection. This proved that GP could be used for IA functions including policy interpretation and enforcement. The researchers did not use typical GP fast Machine Learning in the study. They allowed the GP algorithms to learn slowly as they theorized that minutes or hours of learning time is OK for policy implementations. Typically GP learning takes only seconds. The study showed that GP is unsuitable for unchanging policy; GP was shown to be very well suited for dynamic policy implementation. Learning for static security policy was based on decision sample sets. Over one hundred GP runs were conducted during fitness testing. Exponential GP node growth or bloat conditions were not studied during the research. The researchers observed indications of bloat but made no attempt to solve it and worked around it, which is a weakness of the study. Advanced experimentation in dynamic policy decision making using GP was conducted after static policy evaluation. Policy changes were introduced into the training sets as the Machine Learning continued. Performance was very good for dynamic policy. Policy changes were minimal and prior sample sets included changes.

Elagouni, Garcia & Sebillot (2011) developed a neural network based solution to recognize embedded text in videos. The file format for the videos examined was MPEG based. The research used an automated system for extracting embedded text from videos using Optical Character Recognition (OCR) algorithms combined with an algorithm for predicting the next word in a text segment. Two neural network approaches were used. First a Support Vector Machine (SVM) approach was tested with a recognition rate as high as 81.18% employing 8,544 Support Vectors. The SVM used a Radial Basis Function (RBF) kernel. This is pertinent to the deception detection research contained in

this report because of RBF kernel suitability to the NN network chosen for classification. The researchers compared the SVM approach and results with a Convolutional Networks (ConvNets) approach. Recognition rates for the ConvNets based approach were as high as 98.04% during test runs. The researchers concluded that the ConvNets approach outperformed the SVM approach as is apparent but also noted significance in the lower complexity of the ConvNets approach. Both approaches used 80% of the data sample for training 10% for validation and 10% for testing. A major weakness of this study is that no discussion of the tool used for Neural Network training, validation and testing was evident. This study showed the value of testing multiple Machine Learning techniques during developmental research efforts. Future research identified by the authors includes integrating a speech recognition capability into the system and increased automation.

Azab & Eltoweissy (2011) studied heightened measures of protection from attack as it relates to cyber-physical systems (CPS). CPSs have physical as attributes combined with computing capabilities. A platform was developed by the researchers known as the Cooperative Autonomous Resilient Defense Platform for Cyber-Physical Systems (CyPhyCARD). CyPhyCARD uses biologically inspired computing. The logic within the CyPhyCARD protects a CPS by employing an adaptive, distributed, and geographically dispersed set of components. These components possess dynamic configuration combined with evolutionary intelligence for system attack responses. The design allows for dynamic reconfiguration in response to cyber attacks due to the biologically inspired design. The components are self monitored and are self aware and aware of environmental conditions. They are designed to replicate living organisms. The researchers call each organism-like computing entity a cell. Cells are recruited by other

cells at runtime. Recruitment entails the recruited cell accomplishing goals of the recruiter cells. CyPhyCARD was designed for protection from system attack and to provide protective measures. The human white blood cell activity in the blood stream is an example of how CyPhyCARD works. When an attack occurs on a CPS, cells respond to isolate and disarm the attack similar to how white blood cells travel to the infected tissue of the human body. CyPhyCARD is an advanced technique for protection of distributed sensor systems. The logic built in is pertinent to the deception detection in online data communications research in that it shows how autonomy and adaptability can be built in to help ensure automatic and autonomous information assurance functionality. This has potential for future work extending the proposed research area.

Gondotra, Singhal & Bedi (2011) researched system security with a combined multi-agent and Fuzzy Logic defense in depth strategy for securing systems. The researchers combined autonomous multi-agents and used their decision making plus artificial intelligence. Combining these two capabilities the researchers developed a dynamic security structure adaptable to human behavior modification. Hackers have the ability to alter their approaches to gain access and change methods of attack. This was pertinent to the research reported within this document because after online deception detection becomes the norm, and as deceivers attempt to fool deception detection systems, the system will need to possess dynamic adaptability.

Gondotra et al. (2011) included Fuzzy Logic to compensate for when multi-agent security systems fail and stop running. If the agents do not have a viable solution in their plan library to respond the agents abort. Sophisticated hackers use adaptive attack techniques and multiple attack vectors according to the researchers. This type of dynamic

attack starts with probing the system defense measures and protection implementations. The researchers employed threat models to classify threats by system. This model was used for the basis of the Fuzzy Logic for adaptability categorized by type of threat. The researchers used the model as a layered system defense component asserting that increased defense layers within a system will make it safer it to operate and lower risk of attack.

Three layers are used. Layer one contains single task and goal oriented agents. The second layer uses remediation if layer one is compromised or is deemed not effective. Fuzzy logic is the basis behind remediation and is added to the agents' reasoning ability. This provides more adaptability and acts as a backup for aborted counter threat efforts. The third layer contains meta-agents. Meta-agents contain Fuzzy Logic in a multi-agent approach for reduced attack threat. Meta-agents are able to evaluate the security posture of the system and provide system security agent monitoring reducing the overall threat. Increasingly Information Assurance capabilities are being researched using agents. This type of approach was taken into account when formulating the technical approach taken in the research described in the next section of this report.

Lau, Liao, Kwok, Xu, Xia & Li (2012) explore the use of text mining to detect spam in their research effort. The approach also uses probabilistic reasoning using a Support Vector Machine (SVM). The target of the investigation is false reporting in product reviews. The volume of data in online product reviews is very large and was a challenge for the research team. False reviews are persistent and can remain undetected and undeleted for years. The persistent nature of the false reviews can influence buyer decisions for years. Typically bogus reviewers borrow genuine review material and

modify it to include negative comments. Only slight changes to the genuine review are made. The change usually is in the form of the word "like" being inserted in place of the word "love" yielding a less favorable rating. The research also uses a semantic deception detection operation which searches for missing factors. The research yielded a prototype system or spam in online review detection, but a production system was not initiated. The system has two capabilities to capture data. One capability searches a database of previously stored online reviews. The second capability allows the system to crawl the web to gather online reviews to process. A probabilistic reasoning model analyzes the reviews for spam cues using word replacement detection. The reviews are ranked to indicate the probability that each is spam. Suspicious reviews are displayed to the user graphically. This keeps users from becoming overwhelmed as the data sets are very large. To narrow down the search for bogus reviews, a user of the prototype enters a product description to search for. Two examples of text replacement are examined. One is obfuscation within the body of the review text, the second detects obfuscation within the review title. The SVM determines if semantic overlapping occurs as the method for obfuscation detection.

Santos & Li (2010) conduct deception detection research in knowledge based systems. These systems allow experts to render opinions on scientific and legal matters, which are very narrow subjects. Intelligent agents were used to portray experts rendering opinions within knowledge based systems. The agents were given individual opinions as experts in a narrow field. A Bayesian Network (BN) was used to determine the distance between the nodes within a graph with respect to opinion clusters. This approach was more effective in deception detection than human ability. Two types of deception were

detected. The first classification target was misinformation. Misinformation is described as false expert opinion which is given unintentionally. Misinformation is harmful but not malicious. The expert is considered uninformed or misinformed on the topic. The second type of target was disinformation. Disinformation is the intentional act of misleading by giving false expert opinion. The technique was effective for identifying both types of deception. Agent actions and goals were used to characterize deceptive opinions. Three steps were involved:

1. Activation- Evaluation of the potential deceptive opinion
2. Hypothesis Generation- Determine reasons for the potential deception
3. Hypotheses Evaluation-Ascertain validity of the hypothesis from step 2
4. Global Evaluation- Conglomerate hypotheses and judge the deception

BNs were used to test the agents in a simulated setting of human reasoning. The expert reasoning is encapsulated within the agent for opinion rendering within the BN. The agent's were then perturbed to alter the generated opinion to appear deceptive and cause their opinion to fall further from the cluster of opinions on the map.

The agents' Conditional Probability Table (CPT) was perturbed to simulate deception (Santos & Li, 2010). Perturbing was used to inject uncertainty into the opinion that the agents rendered. The effect was that the opinion was skewed from other opinions within the narrow focus area. Other non perturbed agents rendered opinions which were not skewed. No actual deception was used, only perturbed opinion rendering agents. One assumption of the research was that experts will closely render opinions based on their knowledge and experience in such areas as law and medicine if they are not being deceptive. Therefore within the BN outlying opinion would indicate deception. An

existing BN was used as a test bed. This could be a disadvantage as the effectiveness of the BN might not be established for this test case. It could be an advantage as well since it is a known working BN and the researchers did not have to develop the BN themselves. The researchers were able to detect both misinformation and disinformation from the perturbed agent opinion renderers using the BN with greater accuracy than a human could. The researchers conceded that perturbed agents rendering opinions was not a very accurate or realistic simulation of deception and that the exercise was very limited. However, the study was very useful in taking measures to detect deception in expert systems. This study was pertinent to the research presented in this report as it has some similarities in approach.

Table 1.

*Adaptive Resonance Theory Paradigms and Descriptions (*Filippidis, Russo & Jain, 2010).

ART Paradigms	**Description**
ART1	Self organization. Able to recognize binary patterns.
FUZZY ART	Uses fuzzy set theory computational capabilities combined with ART1.
ART2	Adapted from ART1 to include analog input patterns.
ART2-A	More efficiency attributes than ART2.
ART3	Parallel search, theory testing distributed recognition capability resident in multilevel networks.
ARTMAP	Autonomous learning. Classification of arbitrarily ordered input vectors into recognition categories based on predictive success.
ARTMAP-PI	ARTMAP extension to include probabilistic outcomes.
ARTMAP-DS	Extension of ARTMAP to include input discriminators.
Gaussian ARTMAP	Supervised learning of analogue multi-dimensional maps.
EXACT ART	Includes regulatory logical functions combined with ART.
ART-EMAP	Pattern class recognition after both supervised and unsupervised learning.
MART	Multichannel signal pattern recognition with no prior supervised learning.

Filippidis, Russo & Jain (2010) discuss Adaptive Resonance Theory in their research on ART2, thermal imaging and Multilayer Perceptrons. Contained in Table 1 is an explanation of many ART implementations. ARTMAP proved to be of interested to the deception detection research effort described in this report due to its ability to classify input vectors by category using binary classification. Binary classification was important to the deception detection research because it was used to determine if deception was present of not. Below are some discussions of research efforts using Fuzzy Logic combined with Adaptive Resonance Theory to extend binary classification techniques.

Ross (2011) discusses the fact that closed algebraic formulae cannot process imprecise data that fuzzy systems are designed to process. Fuzzy logic is used to calculate values for imprecise and missing data elements without straining computer processing resources. Fuzzy processing provides grouping of data for analysis and algebraic processing merely provides data. The fuzzy approach allows for undefined data sets to be analyzed yielding approximate solutions. A system that automatically pulls an aircraft out of a nosedive is given as an example of an effective fuzzy solution system. Algebraic approaches would be too time-consuming to correct the situation in time. The algebraic approach would also need specific altitude and attitude values to make calculations. A fuzzy approach uses approximations to make altitude and attitude corrections then level the aircraft flight off. The system would then hand the controls back to the pilot or autopilot. Without fuzz logic development of such a system to correct a fatal nose dive situation in aircraft would not be possible. Fuzzy logic uses a mathematical method to show undetermined and even vague relationships. The concept that uncertainties can be modeled computationally to show condition manipulation results

which are based on uncertain conditions and then calculate predictions of outcomes despite vague or missing inputs is the cornerstone of Fuzzy Logic theory. To simplify, Fuzzy Logic can be expressed by set theory. In contrast to classical set theory, fuzzy theory uses degrees of inclusion in a set as opposed to the classical set theory of binary inclusion. That is, in classical set theory, either a value is included in the set or it is not included. The boundaries of a fuzzy set are blurred or grey areas where the boundaries of set inclusion are not as distinct.

This portion of the literature review is reserved for discussions of research efforts using Fuzzy Logic combined with Adaptive Resonance Theory to extend binary classification techniques. For example, Filippidis et al. (2010) conducted research combining Adaptive Resonance Theory 2 (ART2) implementation coupled with a Multi Layer Perceptron (MLP) function based NN plus thermal imaging techniques with the goal of improved performance of Automatic Target Recognition (ATR) systems. The research successfully extended the outputs of the ART2 MLP function based NN to include fuzziness. The system was able to use binary as well as non-binary input vectors as inputs into a fuzzy fusion layer. The false alarm rate was reduced within the system while the accuracy rate increased. The fuzzified outputs of MLP and ART combined with the thermal imaging analogue outputs were pushed to a fuzzy fusion layer. A special algorithm discriminating background noise was used for the smoothing the thermal imaging outputs. Texture measurements from the target area combined with the Red, Green and Blue (RGB) Infrared (IR) spectral bands were used. Contrast, angular momentum and correlation attributes of the target area were used for inputs into the ART2 NN. A variety of landmines from U.S., Italy and Russia were planted in a test area

according to their specifications for test purposes. Distracters made from manmade materials were also placed within the physical test area. Hockey pucks, PVC material and aluminum objects were among the distracters. The distracters were representative of the materials that landmines are composed of. The results of the research proved that the multi-sensor approach performed better than a single sensor approach because the false alarm rate was reduced and the accuracy increased.

Oong & Isa (2012) researched extending Fuzzy Adaptive Resonance Theory Mapping (ARTMAP) into a multilayer capability. The researchers investigated the belief that it is possible to boost performance during the training and validation process beyond standard Fuzzy ARTMAP by implementing a multilayered approach. An algorithm developed to process Multilayer Fuzzy ARTMAP is described in the article which then describes the experimentation phase. Handwriting samples were examined in one of the experiments. This was pertinent to the deception detection research because handwriting is a form of text. The researchers attempted to determine the handwriting authenticity within media online. The research was an Information Assurance application which paralleled the deception detection research described herein. The result of the research was a decrease in learning time as well as testing time using Multilayered Fuzzy ARTMAP. The researchers found that accuracy decreased slightly (.18%). As the number of training samples increased, the accuracy rates increased correspondingly. The researchers conclude that the larger the training sample size, the lower the impact on testing accuracy using Multilayered Fuzzy ARTMAP. Significant decreases in learning times were observed comparing Multilayered Fuzzy ARTMAP to Fuzzy ARTMAP experimentation results. Although the training sample size increased, the training time

decreased. The researchers found that this was true of evaluation times in correlation to sample size. Overall Multilayer Fuzzy ARTMAP had decreased testing times in comparison to Fuzzy ARTMAP and as the sample size increased, performance improved. This research is pertinent to deception detection research as it shows possibilities to decrease learning and testing times as the example training inputs increase if necessary.

Machorro-Fernandez, Para-Vega & Lopez-Juarez (2010) studied training people with a technique to extend simple ARTMAP into Fuzzy ARTMAP. A mouse was used as a design implement. Readings were captured as input vectors. The hand motion of the experts completing tasks was used as the basis for the training samples. An ARTMAP tool was fed the readings from the expert samples as input vectors. Primary classifications were output by ARTMAP. Then the ARTMAP generated primary classifications were fed into Fuzzy ARTMAP which was trained on the haptic interface (mouse) movements. In the training environment the Fuzzy ARTMAP was used as the trainer for non-expert designers. Fuzzy ARTMAP examined the non-expert trainee attempts at certain design implementations and offered corrections based on a comparison of trainee inputs to stored expert designer haptic inputs. Calligraphy was used as the design medium for the experimentation phase. Fuzzy ARTMAP corrections were used by the humans as the experiments were conducted. The abilities for designing calligraphy based text artifacts increased over time for the trainees. This phenomenon proved that the untrained could learn skills from a Fuzzy ARTMAP trainer according to the researchers. Robotic training is planned for the next phase of the Fuzzy ARTMAP research to correct a robot in the robotic skill building implementations. This article is

relevant to deception detection research because it illustrates how the training portion of ARTMAP can be applied to textual based inputs.

This literature review supports the research presented in this report because it illustrates how deception detection research efforts have identified indicators of deception contained within text for analysis. It also shows the evolution of research into single purpose deception detection tools. Those tools usually require training for users to operate them and are not typically integrated into the system where the data is stored. Suspected deceptive text is usually extracted manually and entered into a deception detection tool. Artificial Intelligence has been advancing toward automated IA tools, but these tools are based on policy enforcement. In one case intelligent agents were employed on a deception detection system but were not used to detect deception. Instead they were used to render opinions and certain agents were perturbed to appear as if they were rendering deceptive opinions. The perturbed opinions were combined with unperturbed opinions and processed by a BN. The BN considered the probability that the outlying opinions were deceptive. Although the evaluation of the capability proved effective in detecting simulated deception in the form of misinformation and disinformation, no real deceptive data was used.

Chapter 3

Methodology

Overview of the Research Methodology

Developmental research was chosen for this deception detection prototype system creation research effort. Developmental research was proven to be the best approach for this type of study based on the capability usefulness that resulted in the system which was developed in the short amount of time allotted to the conduct of the study. The selected approach was not novel as BDI agent research is commonly carried out using a developmental research methodology. This assists in developing capability quickly with a high degree of functionality (Ma, Chen, BAi, & Huang, 2010).

BDI agents were a design consideration that proved critical in the creation of the system prototype. This proved true in part because objects developed with Object Orient Programming (OOP) require that methods are triggered by humans that must accomplish a task. In contrast a BDI agent within the prototype autonomously executed some of the methods with no human interaction. For example, the BDI agent Message Event methods (BDIMessageEvent) were used to initiate and synchronize agent plans containing activities. This was critical in BDI agent interaction with the online database communications system back-end database. To illustrate the importance of this, consider that if the timing were off in a database related task, a table might not have been accessible to accept NN input vector data resulting in an empty table, lost or corrupt data and potentially system crashes. BDI agents usage was critical for the timing and synchronization of tasks to avoid these situations. BDI agents can sense their

environment if their belief system contains environmental condition statements. This gives the BDI agent a sensor advantage over other approaches because the BDI agent belief system can contain knowledge of the system and network environment. The BDI agents sense changes in the environment and execute tasks in response. This posed an advantage over other approaches which were considered. In building the prototype BDI agent Goal Event Methods (BDIGoalEvent) allowed the agents to achieve goals using meta-level reasoning and respond to conditions in the environment (Agent Oriented Software, 2011). BDI agents can detect and respond to changes in their environment due to reasoning methods built in to the BDI agent class library as well (Ma, Chen, Bai, & Huang, 2010). These reasoning methods allow the BDI agent to execute actions based on human behavior models and are built in to the plan library. The plan based reasoning is considered the BDI agent top-level reasoning method because it is built into the plan body. For a practical example, in place of a database administrator querying a database, consider a BDI agent completing the action modeling human behavior. Another advantage of the BDI agent capability was the ability of this type of agent to successfully initiate NN functionality by launching the NN interface as a module after input vector creation. Combining BDI agents and NN functionality is not a novel concept. This concept was proven in other research and was highlighted in the research of Kuderna, Hoszu Vacariu & Cret (2009), which is addressed in the Literature Review.

The methodology employed during this research centered on the creation and evaluation of BDI agent automation and effectiveness and NN classification accuracy within a deception detection prototype. BDI agents proved very useful operating within

the prototype due to BDI agent autonomy plus BDI agent software based FSM synchronization attributes (Ma, Chen, Bai, & Huang, 2010).

As stated, the prototype designed under this research effort was successfully completed using a developmental research methodology. The design was validated using a testing process to determine the level of performance for the system. Autonomous BDI agent task execution performance metrics and NN input vectors extracted from online data communications within a Weblog server were used as the units of analysis. No Personally Identifiable Information (PII) was entered by any data provider and no PII was used in any writings or artifacts submitted or published during the research effort. Raw data from participants providing the data was not included as part of this report, but is available upon request.

During the course of this research a prototype deception detection system was developed. The BDI agent modules were created using the Java programming language and JACK agent development IDE. The NN portion of this research effort was completed using WEKA software. The prototype under development was evaluated on how well it performed automated deception detection tasks using BDI agents and how accurately the NN classified the input vectors as either “deceptive” or “not deceptive”. The prototype was integrated directly into an Apache Roller Weblog server which replicates real-world online data communications environments. The BDI agents interacted with the Roller Weblog server and the backend database and were able to extract “weblogentry” table contents for processing and passing output as usable input vectors. Developmental research proved effective in accomplishing all of the tasks required for deception detection prototype development and was the correct methodology under the conditions

chosen. This research method was chosen after evaluation of other BDI agent research in the literature highlighted its effectiveness. The developmental research methodology is commonly employed during BDI agent research efforts (Ma, Chen, Bai, & Huang, 2010).

Specifics of the Research

This portion of the deception detection research report will delineate the specific steps of the research, how the prototype operates, the data collection process and the testing of the prototype. This section is as detailed as possible to accurately describe how the system was built, how it works and how it was tested.

System Environment Setup

The developmental research methodology is quite effective when development takes place within an environment that replicates a real world situation. For this reason Apache Roller Weblog server was chosen as the platform that the prototype would be integrated into. This is an open source Weblog server environment which operates using a Java Virtual Machine (JVM) and has options for several Java Servlet containers for access to the web application. For this implementation Apache Roller 5.0.1 was used. The system build requires that the Java Development Kit (JDK) be loaded onto the platform. JDK version 7 was obtained as an open source download and installed on the system in preparation. The Apache Tomcat version 6.0 Servlet container was selected for the build. This is also an open source software package which acts as the Web interface and Java Servlet container for the Roller Weblog server access. This was obtained, downloaded and configured in accordance with the standard Tomcat 6.0 installation documentation. The Apache Roller Weblog server required the MySQL 5.1 open source database application for usage as the backend data container. This was obtained and

installed in accordance with the MySQL 5.1 installation and configuration instructions. The Apache Roller Weblog server requires the usage of the Java Database Connectivity (JDBC) API specification for database connectivity and operations on the MySQL database. The JDBC version 3.1.1.4 was downloaded and placed within the appropriate Tomcat file folder in accordance with the Apache Roller Weblog installation instructions. Apache Roller Weblog server was then download and configured using a standard Tomcat "WAR" file deployment (Apache Tomcat 7, 2011). Once the Roller Weblog server was installed, configured and operational a decomposition effort was undertaken. For the integration of the system prototype there was a need to discover how the Apache Roller Weblog server interacted with the MySQL database as this would be the model for BDI agent database interaction. The research required that only the data needed for input vector creation and user tracking would be necessary for the BDI agents to extract. This would also cause minimum impact to the operating environment. It was determined that the Roller Weblog server contained a single database named "rollerdb" (Apache Roller, 2011). The table of interest containing the user data and the text for the Weblog content were contained in a single table named "weblogentry".

The research now had a need to modify the "rollerdb" database by creating a table named "vectors". This would contain the user name of the Weblog entry as well as the input vector outputs generated by the prototype. It was important to ensure that this had no impact on the Roller Weblog server operation. The "vectors" table was created using the JDBC API database specification successfully. This was accomplished by creating a Java program containing SQL commands to create the "vectors" table by running the program. This was a major milestone in the development process because during the table

creation process, no impact to the Roller Weblog server environment was observed. It was also important because it would serve as a model for the Java module inclusion in BDI agent plan libraries for interaction with the MySQL database using the JDBC API.

Once the operational environment was setup, it was tested for functionality in preparation for prototype integration. A method for Weblog text entry by data provider volunteers was also devised to ensure the data was available and could be easily located within the "weblogentry" table of the "rollerdb" database. No issues were uncovered during evaluation of the Roller Weblog server operational environment. Once the evaluation was complete, the research moved into development of the deception detection prototype.

BDI Agent Functionality

BDI agent functionality was designed to be the backbone of this very practical deception detection prototype system design. An online data communications deception detection prototype was created by implementing a Java based, BDI agent and NN modular approach. The prototype was designed with BDI agent integration considerations for the online data communications environment, the data contained within its database, and ability to create usable deception detection input vectors for input into a NN (Toma & Hancock 2010). This was highlighted in the system environment setup description. Linguistic cues to deception were processed into NN input vectors using algorithms contained in the executable code embedded into BDI agent plan libraries. The BDI agent desires drove the agent toward the goal of executing Java code that queried the Roller Weblog server backend "rollerdb" database and converted the content into input vectors usable for NN classification as "deceptive" or "not deceptive".

The BDI agent intentions contained in the plan library hold Java executables that allowed for the BDI agents to automate many of the deception detection tasks including file and database maintenance operations. The plan libraries hold modules that facilitate the creation of the input vectors based on linguistic cues to detection for further processing by the NN. Input vector generation algorithms were designed to allow raw Weblog server text to be transformed into input vectors. The algorithms created are very basic text parsing algorithms which search for deceptive characteristics within text. The following describes the input vector generation algorithms. These were created using Java standard text manipulation methods to extract characteristics from the "weblogentry" database table of the "rollerdb" database. Although seven of these input vector generation algorithms were based on the theory of linguistic cues to deception as described by Zhou & Zhang (2008), other linguistic cues to deception observed within the Weblog text allowed for the creation of three more algorithms. How this came about is described below:

1. After examination of text within the Weblog server entries it was determined that differences existed in the way individuals employ comparisons when being deception versus being truthful. An input vector generation algorithm to determine the types of comparison words was developed to capture these as input vectors.
2. During analysis of Weblog server entry raw data it was uncovered that deceptive Weblog server users describe the positions of objects differently than truthful Weblog server entry creators. An input vector generation algorithm was developed to capture these differences in positional depictions of objects.

3. Further analysis of the Weblog server raw data entries revealed that the way deceptive Weblog server users use analytical wording differs from analytical wording of truthful Weblog server users. An input vector generation algorithm was developed to extract input vectors for analytical differences.

The following is a description of all ten of the input vector generation algorithms and how they operate:

1. Word Count algorithm is a basic Java method of cycling through the TEXT field of the "weblogentry" database table in the Roller database. The total number of words are determined and placed into a variable. The system accesses the "vectors" database table and places the value of the variable into the word count column of the table. This is used as the word count input vector.
2. Word complexity is calculated using a basic character counting Java method. The total characters in the TEXT field of the "weblogentry" database are determined for each entry. This value is stored in a variable which is divided by the word count variable. The result is stored as the complexity variable and stored in the "vectors" database table as the complexity input vector.
3. Self References are calculated by using the java.util.regex package and in particular the regionMatches() method member to cycle through the "weblogentry" database table TEXT field to search for instances of references to the writer of the entry. The individual references encountered are incremented. The sum of the self references encountered are placed into a variable which is written to the "vectors" database table in the Roller Weblog server back-end. This entry is used as the input vector for self references.

4. The motion input vector generation algorithm uses the java.util.regex package and its associated Java regionMatches() method to cycle through the “weblogentry” database table’s “TEXT” field for instances of motion words. When a match is found a counter variable increments to count the amount of motion words found. The counts of all motion words encountered is summed. The sum of the motion words is placed into a motion vector variable and placed into the “vectors” table of the Roller MySQL database within a column containing the motion input vector entries.
5. Sentence completion is determined by cycling through the TEXT field of the “weblogentry” database of the Roller Weblog server using the java.util.regex package member method regionMatches(). The algorithm searches for punctuation that indicate full sentences have been used as opposed to fragments. These are placed into variables. These variables are summed and placed into a sentence completion variable. This variable is entered into the “vectors” database table as the sentence completion input vector for the result set.
6. Adjective and adverb descriptive terms are extracted by cycling through the TEXT field of the “weblogentry” table of the “rollerdb” database. The java.util.regex package and regionMatches() method is again used to search. This algorithm identifies adjectival and adverb endings. The instances of those identified are added together and placed into a variable which is entered into the adjective adverb column of the “vectors” database table and used as the associated input vector.

7. Word uniqueness is determined using the java.util.Scanner object. The TEXT field of the “weblogentry” database table from the Roller Weblog server is run through the Scanner object. The algorithm compares words to determine if they are used more than once. If the words are not used more than once, they are determined to be unique for the entry and placed into a HashMap object. The keys of the HashMap are placed into an array. Once in the array the entries of the array are summed. The sum is stored in a variable which is written to the “vectors” database table on the Roller Weblog server as the uniqueness input vector.
8. Comparative content terms are collected by iterating through the TEXT field of the “weblogentry” table contained in the “rollerdb” database. The java.util.regex package and regionMatches() method is once more employed to search. This algorithm identifies terms which the user might use to compare items that they are writing about. The instances of those identified are summed together and placed into variables. These variables are then summed and the sum is entered into the comparative column of the “vectors” database table as the associated input vector.
9. Object position terms are identified by processing the TEXT field just as the majority of the of the other vector generation algorithms using java.util.regex and its regionMatches() method. The results are placed into the “vectors” table as the position input vector. The terms identified are related to the position descriptions of objects that a Weblog creator would use to describe how objects are laid out from their viewpoint.
10. Analytical content terms are searched using the java.util.regex package and the regionMatches() methods just as many other algorithms employ it. The matches

are summed and the total for a Weblog entry are entered into the "vectors" table as analysis input vectors.

Creation of the input vectors was based on basic calculations linked to various characteristics of the text within the Weblog entries. It was determined that a basic approach for calculation was best in the initial prototype development effort to reduce system processing due to the potentially large amount of data that would need to be processed and the complexity of interfacing with a Relational Database Management System (RDBMS) using the JDBC API.

System Functional Description

Figure 1 depicts the functionality of the system architecturally. The depiction shows that the prototype system was built into an Apache Roller Weblog server (Apache Roller Weblogger Version 5.0, 2011). As Figure 1 shows, the system was built using a JVM. This was an important design consideration since BDI agent programming languages are actual extensions of the Java software language and fully support Java objects. This allowed portability, flexibility and functionality as complete Java libraries were available during development and BDI agent based modules were able to incorporate all Java objects. The prototype includes Apache Tomcat Server to provide the web based Servlet engine for the Apache Roller Weblog server to be deployed as a web service (Schildt, 2011). When a user enters data, it is stored in a MySQL database. The Tomcat Servlet container uses a JDBC API specification connection to communicate with the Weblog server backend database. All user created content was stored in the MySQL database. BDI agent based interaction with the MySQL database was modeled after human database interaction behavior. The BDI agent model was used to automate

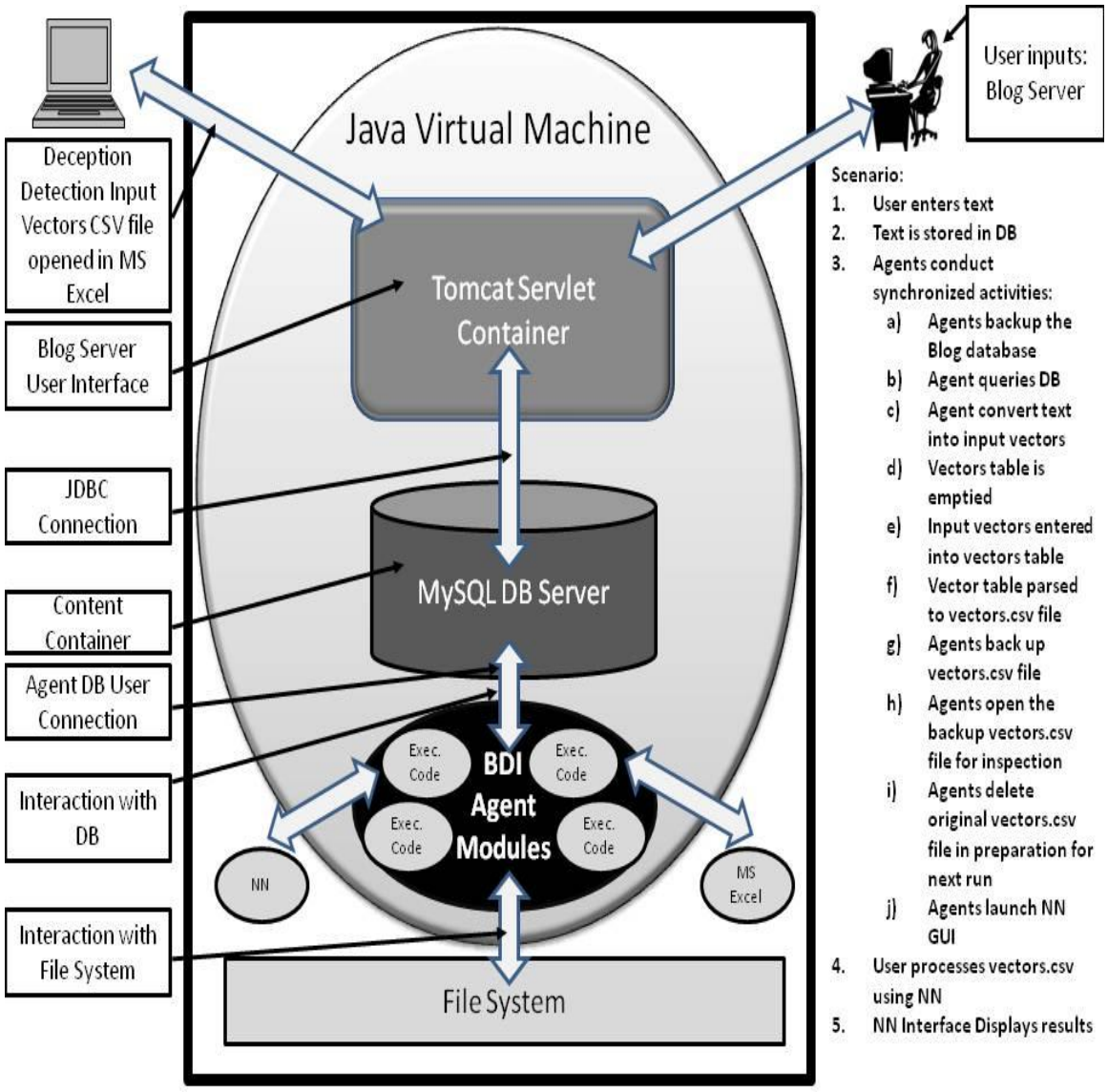

Figure 1. Prototype Architecture and Working Scenario.

deception detection tasks and execute queries against the database with user credentials and privileges. The BDI agent modules extract data and perform table updates from the Roller Weblog server "rollerdb" table by employing standard Data Manipulation Language (DML) and Structured Query Language (SQL) commands contained within prepared statements. The JDBC API is used to open a connection between the JVM containing the BDI agent Modules and the MySQL database.

The following is a description of how the BDI agent based modules operate functionally. The BDI agent module backs up the Roller Weblog Server "weblogentry" database table. When complete the agents notify the deception detection user with another jOptionPane dialog box that the database is backed up. Then the BDI agent modules execute a plan library routine containing a prepared statement to connect to the database table that contains input vector data. To ensure no replicated input vector data entries are carried over, all data within the "vectors" table is removed at the start of execution. This is done because no vector data can be duplicated or it will invalidate the training, validation and testing during the NN operation on the classification of input vectors.

After the vectors table is empty, the BDI agent module queries the MySQL database and extracts the user identification and the Weblog server text entries from the "weblogentry" database table. To do so the agents use the JDBC API to access the database, run the queries against all entries using the ResultSet Java interface. The Java next() method iterate through each user entry one at a time. The text portion of the "weblogentry" Roller Weblog Server table is the main target of the BDI agent queries. Once a result set is retrieved by the agent module the text portion is processed through the input vector generation algorithms which work to extract input vectors from the text as previously delineated. As described, many of the algorithms use the Java java.util.regex pattern matching package utilities. The java.util.regex package is similar to Perl scripting in that it incorporates text retrieval, parsing, search and replace plus other functions common to text parsing languages. Other basic Java text manipulation and search techniques were also used as described. The algorithms are used to extract the

text attributes from each entry one by one and place the values of the operations into variables. These variables correspond to input vector column entries in the “vectors” database table. As discussed in the system environment setup portion of this document, the Roller Weblog server “rollerdb” database had been modified to include the creation of the “vectors” table previously. As the algorithms finish processing the input vector data for an entry, the BDI agent modules store the entry by updating the “vectors” database table. The user name is also stored for future development purposes and tracking. The user name is not linked to any of the data providers and contains no PII.

Once the BDI agent processing of all of the text entries into the “vectors” database table is complete maintenance and file operations are performed. First the query of the “vectors” database table and extraction of the vector data occurs. Then the input vector data is placed it into a Comma Separated Value (CSV) file named “vectors.csv”. Once this is complete the BDI agent module backs up the CSV file by copying the original “vectors.csv” file and renaming it using a date time stamp as a unique identifier. Once the backup file containing the input vectors is created, the file is then opened in MS Excel by the prototype for the deception detection system user to inspect. Upon completion the BDI agent module notifies the user using a standard Java jOptionPane dialog box that the operation is complete.

When working with databases and BDI agents it is necessary to ensure operations are done in sequence. To accomplish this, the prototype design incorporated the use of the aos.jack.util.thread.semaphore synchronization resource. This allows the use of a First In First Out (FIFO) sequencing of activities (Agent Oriented Software, 2011). The BDI agents take control of the processing thread by grabbing it using a method embedded in

the plan library. To grab the thread the planWait() method is used. The planWait return is a special JACK return type known as a cursor. This is similar to a database cursor which allows cycling through database records using the resultNext method. For JACK implementations the cursor returns are considered triggered cursors because they are event driven. The triggered cursors event trigger is the BDI agent plan grabbing the processing thread and is initiated by the planWait method which returns the cursor. To make sure the timing is correct the BDI agent has an @waitFor reasoning method within the body of its plan library. The executable code will not run until the @waitFor reasoning method is used. With the semaphore based synchronization, the BDI agent @waitFor reasoning method is waiting for the processing thread to be free. Using this method it is possible for a BDI agent to synchronize activities by having the @waitFor reasoning methods contained in its plan. The BDI agent plan can the send messages to synchronize events in the order that is necessary. The plan uses the @subtask method to execute tasks in order. Once the task is complete the plan in the BDI agent plan library releases the program thread and uses the @send method to indicate the completion of the task and indicate that the processing thread is available. Once the @send method is complete the BDI agent plan processes an @waitFor method once again to wait for thread grabbing opportunities in response to BDIMessageEvent events. The description of the activities of the BDI agent synchronization above actually creates a BDI agent software based FSM (Agent Oriented Software, 2011).

The Java based executables were incorporated into BDI agent plans. Other Java executables were called from the main Java class file where appropriate. Allowing the BDI agents interaction with the database system gave the prototype flexibility and kept

the prototype in a low coupled state. Low coupling added flexibility in the design for the system. During the design phase the low coupled state design was quite beneficial. For example, if a BDI agent based Java module was modified, the agent interacting with the executable did not need to have its design modified. The executable code within the BDI agent module operation in Figure 1 represents the idea that a BDI agent executes all tasks using Java utilities. The BDI agent module launched the WEKA NN tool which was used to classify input vectors generated by the system (Hall, Eibe, Holmes, Pfahringer, Reutemann, & Witten, 2009). The NN trained using predictor variables extracted from the input vectors CSV file. It is important to note that the BDI agent architecture allowed for interaction with the Operating System (OS) and the file system that it managed. This is accomplished using standard input-output (IO) operations using standard Java file operations methods. Doing so allowed the system to produce usable input vectors in the format acceptable for the NN interface.

Deception Detection System Operation

Figure 1 illustrates how the system operates in sequence from start to completion. The system's BDI agent software based FSM synchronization and BDI agent autonomous nature allowed for the rapid working of tasks that humans would do very inefficiently. The steps outlined below characterize the operation of the system prototype. The steps are carried out by BDI agent based modules where indicated. Every effort was made in the development of the prototype for creation of BDI agent autonomous and synchronized activity:

1. The user enters his or her username and password in the Weblog server login page to gain access.

2. The user enters text and saves it into the backend database using the Roller Web based GUI.
3. The BDI agent modules conduct autonomous and synchronized activities
    a. The database table (“vectors”) described in the system environment setup and created with the intent of containing input vectors for every run of the application is emptied of its contents in preparation for new input vector storage.
    b. BDI agent modules backup the Weblog server database.
    c. BDI agents login to the database using the JDBC connection API and run prepared statement queries on the Roller Weblog server ”rollerdb” database “weblogentry” table.
    d. The BDI agents cycle through the query results and run the text entered by each user through a series of ten input vector generation algorithms. As described in the BDI agent functionality section, these algorithms extract linguistic indicators of deception from the user Weblog text and convert the text into input vectors (Zhou & Zhang, 2008).
    e. The BDI agent modules process the output of the input vector generation algorithms and then store them in the “vectors” database table by using the JBDC API and a prepared statement executing an insertSQL and executeUpdate JDBC commands. The JDBC connection is then closed.
    f. Another JDBC database connection is opened to the database and the BDI agent modules query the “vectors” table using JDBC prepared statements.

The results of this table query are written to a file which has been created to contain the values. This is a CSV file named “vectors.csv”.

g. The BDI agent modules backup the “vectors.csv” file by copying it and renaming it with a date stamp appended to the “vectors” filename. This step must be done because the system will need to delete the “vectors.csv” file in order to execute again.

h. The backed up copy of the “vectors.csv” file is opened by the BDI agent modules for the user to inspect. MS Excel is launched by the system. Excel was selected for this task due to its Graphical User Interface (GUI) and spreadsheet functionality.

i. BDI agent modules delete the original “vectors.csv” file in preparation of another program run.

j. The BDI agents launch the NN GUI for processing of the CSV file.

4. The user inputs the file containing the input vectors using the NN user interface.

The user processes the “vectors” file content using the NN GUI which returns the deception detection classification outputs after training, validation and testing of the NN. The user must perform file operations and operate the NN software as is described in the NN training, validation and testing portions of this report.

FSM Operation

The following is a description of how the BDI agent software based FSM operates with respect to the deception detection prototype system. Two BDI agents are used in the FSM operation. The following steps are broken down for clarity:

1. The program starts and within the main Java code is a module that conducts a Continuity of Operations (COOP) task to ensure the integrity of the data prior to manipulation of the “rollerdb” database. This particular code backs up the Roller Weblog server “rollerdb” database “weblogentry” table.
2. BDI agent one initiates the FSM by sending a BDIMessageEvent to BDI agent two. Upon receiving the BDIMessageEvent, BDI agent two searches its plan library for a plan to post the event internally. BDI agent two uses a #handles event within its belief set for the particular BDIMessageEvent that it received. It also possesses a #posts event allowing it to execute code in its plan library to handle the BDIMessageEvent internally. The plan library also contains the #uses event for the use of the semaphore. BDI agent two locates the code in the plan library allowing it to grab the processing thread using the semaphore. The program will not proceed until the thread has been grabbed because prior to the BDI agent receiving the BDIMessageEvent an @waitFor reasoning method is in the plan library ensuring that the agent is in a wait state until the message event is received. This means that the execution waits until the semaphore has been used to take control of the processing thread. Once the processing thread is grabbed, BDI agent two executes the code associated with the BDIMessageEvent. This code in particular is the module that deletes all rows in the “vectors” table of the Roller Weblog server “rollerdb” database. When BDI agent two has completed the tasks in its plan corresponding to the BDIMessageEvent, it releases the processing thread and sends a BDIMessageEvent indicating that the thread is now available.

3. BDI agent one receives the BDIMessageEvent and looks at its plan library to find executables to respond to the message. It has a #handles event and a #posts event as well as a #send event. It handles the message, posts internally and using the #send event sending another BDIMessageEvent indicating that the processing thread is free. This message has a corresponding event in BDI agent number two's plan library.
4. BDI agent two receives the BDIMessageEvent and again looks in its plan library because it has a #handles event for the message and a #posts event to look in the plan library for code corresponding to the BDIMessageEvent it has received. It locates the plan associated with the BDIMessageEvent that it has received and searches the corresponding code contained in the plan body and finds the appropriate executables. This particular executable code is the main deception detection system code. This part of the program accesses the Roller Weblog server "rollerdb" database and the "weblogentry" and the "vectors" tables as necessary to generate input vectors. This plan library executable also populates the "vectors" table, creates the CSV file, backs up the CSV file and deletes the original CSV file. After the file operations are complete the code in the plan library starts MS Excel, opening the backup CSV file and starts the NN user interface. Initially this BDI aget plan also uses the @waitFor reasoning method to cause the BDI agent to wait for the BDIMessageEvent and then grab the processing thread at the right time employing the #uses event and the semaphor. When BDI agent two has finished executing the code in its plan library it releases

the thread and sends a BDIMessageEvent. This message indicates that the task is complete and the processing thread is available.

5. BDI agent one handles the BDIMessageEvent by finding code in its plan library which ends the program.

Neural Network Validation

A Baseline determination was conducted prior to the comparison of linguistic cues to deception against the baseline. The baseline determination was conducted as outlined below:

1. Data from a group of users was entered into the Weblog server. The data collection was asynchronous. Data providers did not interact with one another and no data was shared. The true user identity was not and will not be revealed during any portion of this study.
2. The users were given instructions on entering truthful entries into the Weblog interface. This was phase one of the process. Appendix A contains the instruction for user truthful data entry.
3. The users were then given instructions on entering deceptive data into the Weblog interface using another login. Appendix A also contains the instruction for user deceptive data entry.
4. The NN implementation trained using 80% of the samples.
5. The NN validated the NN training algorithm using the remaining 20% of the sample which was held back during the training phase. An 80-20 percent training to validation split is an acceptable sample breakdown for validation of NN training algorithms (Whitten, Eibe & Hall, 2011).

When a certain degree of accuracy was obtained during NN validation the research effort moved into a testing phase. To minimize the risk that the NN would become over familiarized with the data with the data during training and render skewed accuracy calculations during testing, mitigation measures were taken to eliminate the possibility. A plan was devised to ensure that the NN environment used during the validation phase was configured on a separate testing computer. The concept was to use a dedicated testing computer system that was physically separated from the computing environment used in the validation phase thereby assuring no NN exposure to the data prior to the testing phase. After validation, the NN training algorithm was discarded to ensure it would not be used for testing. The WEKA NN software was loaded onto the separate testing computer and configured in the same way as the original.

Prototype Development Considerations

Java software modules were tested individually in the live environment where they were expected to operate and then ported to the JACK BDI agent development environment for module integration and further evaluation. For the online data communications server environment, the BDI agent modules were operating using the backend database contained in the online data communications server. During execution the BDI agents interacted with the Roller Weblog server by executing queries as well as updating tables using the JDBC database connectivity API. This approach added flexibility to the system as the JDBC specification is applicable to many commercially available and open source RDBMSs. The design feature gives the prototype flexibility, portability and adaptability to many deception detection scenarios with virtually unlimited data applications. Data about the BDI agent's activities was recorded during a

developmental test. The developmental testing gauged the effectiveness of the prototype's ability to extract deception detection input vectors from an online data communications server and conduct other autonomous and synchronized prototype processing, storage and input output procedures. The modules were tested individually as is the norm for BDI agent development efforts (Ma, Chen, Bai, & Huang, 2010). The system was analyzed to determine how rapidly it automatically generated detection deception input vectors and how accurately it performed NN classification. The NN classification accuracy results were compared to the known probability of human ability to detect deception in written communications which is about 50% according to Zhou & Zhang (2008).

Although a module recoding strategy was adopted for this effort, there was no need to recode the BDI agent modules due to failure to exceed deception detection capabilities (Zhou & Zhang, 2008). This was due to the chosen approach of integrating working Java modules into the AOS JACK development environment. BDI agents proved extremely flexible for this deception detection effort and in particular with regard to the synchronization of autonomous activities due to the agents' belief set construction (Patel & Hexmoor, 2009). The ability to change the BDI agents' belief set has advantages for developmental research in that it gives them flexibility in choice of courses of action and for synchronization within their plan library. This flexible programming ability had a further advantage in that the BDI agents developed for deception detection in online data communications can easily be modified to operate within other online environments.

During prototype development, each time a BDI agent module was modified and NN input vector generation algorithms modified they were reevaluated rigorously. An

individual test plan mapped back to the prototype's overall capability expectations was used to ensure that the testing did not vary (Ma, Chen, Bai, & Huang, 2010). Figure 2 shows the specifics of the developmental research process used in the BDI agent based deception detection system prototype creation. Figure 2 shows how interrelationships were identified during development, design and testing for supporting prototype evolutional phases and how performance improvement redesign considerations fit in to the cycle.

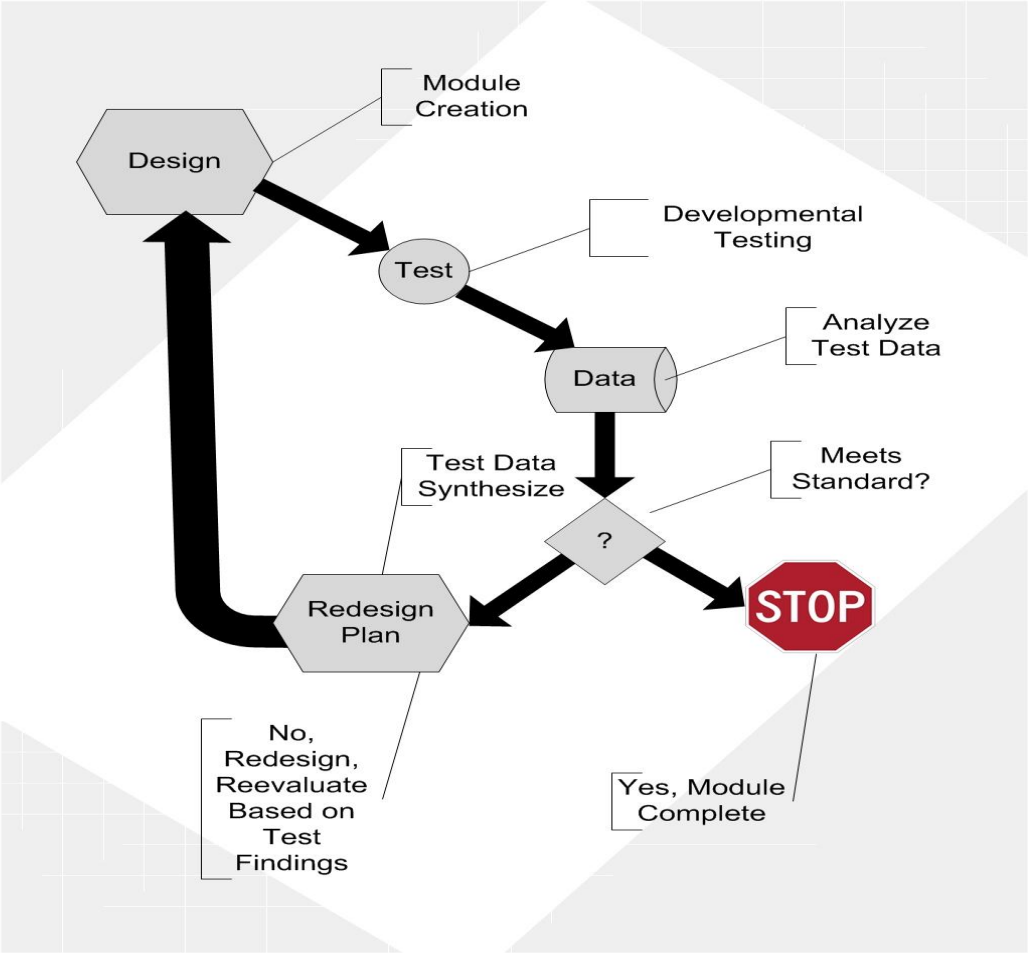

Figure 2. Depiction of the Prototype Development, Design & Test Process.

Classification Considerations

The following Machine Learning software packages were integrated as modules into the deception detection prototype system during the early phases of development for the purpose of evaluation:

- ENCOG3
- MATLAB Neural Network Toolbox
- DTREG
- WEKA ARTMAP Implementation

The rationale of the integration was twofold of all of these Machine Learning software suites into the prototype. First the deception detection prototype design was verified for flexible integration of Machine Learning applications. Flexibility is important for further research. Secondly, each was evaluated based on its ease of use, support documentation, and effectiveness. The WEKA ARTMAP implementation was selected due to its ease of use, Java source code availability, easy to understand output during training and testing runs and in particular allowance for a wide selection of kernel functions to be used for training the NN (Hsiao, 2006). The RBF kernel function was chosen for the NN classification training due to it suitability for two category classifications. The RBF Kernel was easily selected within the NN GUI. The WEKA NN application suite is very well documented with online support documentation, development forums and up to data publications (Whitten, Eibe, & Hall, 2011).
The WEKA documentation contains excellent directions for selection of kernel functions plus descriptions of their characteristics and usage. WEKA also accepts CSV file format for Training and Validation. WEKA has a built in process for separating training input vector sample files from testing input vector sample input files. This is important to

verify that the training algorithm was not exposed to the test data during the NN training phase.

The deception detection system prototype development effort incorporated the NN software for classification. The NN output showed a classification of "T" or "F" as targets for the classification. A "T" classification was shorthand for a "not deceptive" finding and an "F" classification was shorthand for a "deceptive" classification finding. In determining the total True Positives, True Negatives, False Positives and False Negatives, the study relied on the confusion matrix which was provided as part of the output from the NN input vector classification test runs (Kolo, 2010). Figure 3 shows that in the first equation True Positives are added to True Negatives. Then the sum of the True Positives added to the True Negatives would be divided by the sum of deception detection evaluation phase samples attempted by the NN implementation subsystem. This basic math allowed the NN subsystem to calculate a simple binary classification accuracy percentage during the validation and testing phases of the testing cycle. Similarly the NN subsystem classified False Discovery by summing False Positives and True Positives, using this sum as the denominator and using the sum of False Positives as the numerator during test runs. The quotient of this formula revealed a False Positive rate. The formula is also shown in Figure 3. The results of the binary classification validation and test runs will be presented in the Findings portion of this report.

Format for Results

The findings of the research were formatted to show the results of the system performance testing and deception detection classifications using the subsystem outputs. The results are contained within this standard Dissertation Report which conforms to the

Nova Southeastern University Graduate School of Computer and Information Sciences (GSCIS) Dissertation Guide.

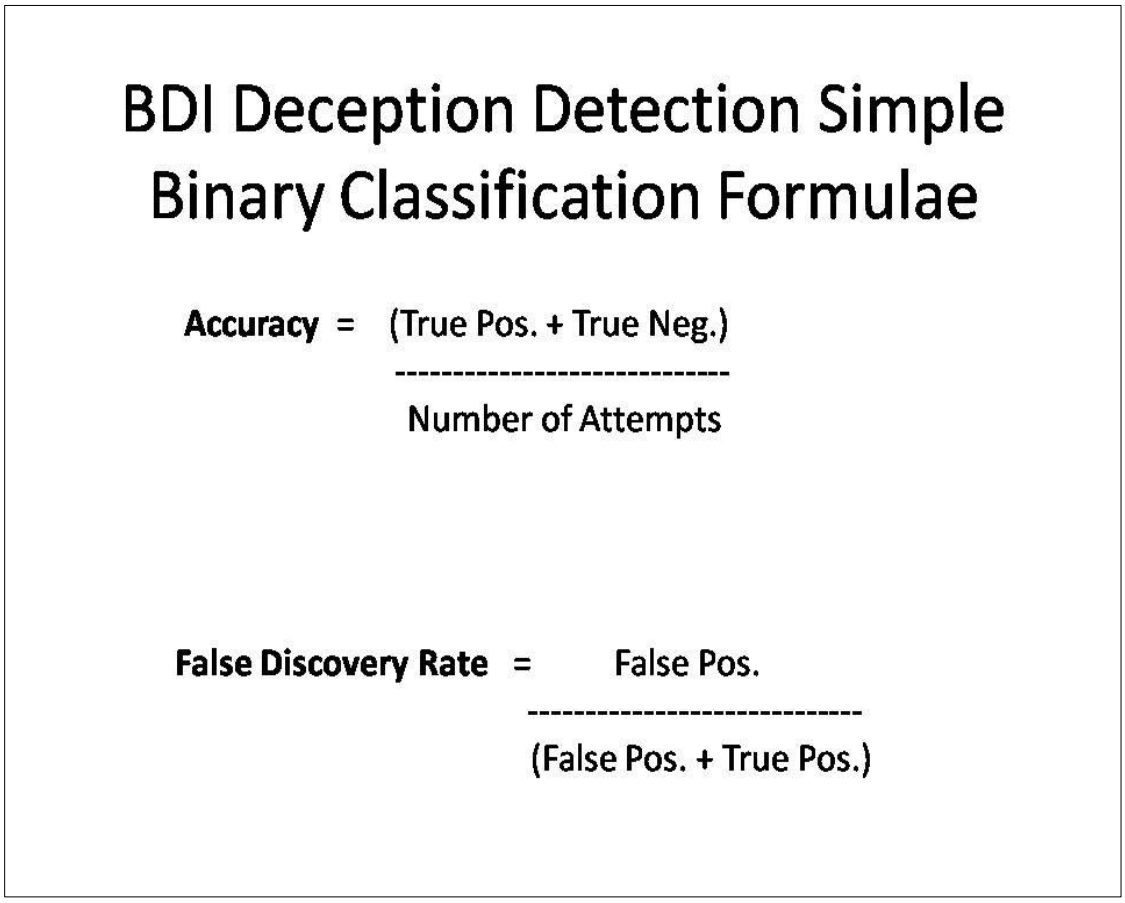

Figure 3. Formulae for Deception Detection Accuracy and False Discovery

Resource Requirements

Certain resources were necessary to accomplish this research. They are listed below:

a. Two laptop computers running MS Windows 8

b. Open source online data communications server software: Apache Roller, MySQL, Tomcat, and NetBeans Java Development IDE.

c. BDI agent Integrated Development Environment: JACK Intelligent Agent Programming Language.
d. WEKA ARTMAP implementation.

All necessary resources were provided by the researcher at his expense.

Summary

This chapter of the research delineates the processes, procedures and methods which were employed in a BDI agent based approach to deception detection in online data communications. All of the steps necessary were explained fully to ensure repeatability of the study. A developmental research methodology was used and was determined to be the most appropriate method given the challenges undertaken, BDI agent automation capability creation, research constraints and limitations. RAD principles were used in this developmental research project. RAD was chosen with the promise of increased probability of success. This eclectic approach reduced schedule risk by reducing the research goal achievement time (Agarwal, Gupta, & Tayal, 2009).

Chapter 4

Results

Data Analysis

The deception detection system prototype was evaluated for performance. This was conducted by taking time measurements on various BDI agent based modules which were deemed critical to system performance. Another performance measurement was derived from the NN classification accuracy. The NN classification accuracy testing was derived from two phases. The validation phase merely allowed the developer to get a feel for when NN classification accuracy was adequate for moving into a testing phase. The testing phase was a formal testing phase with great care taken to protect the integrity of the input vector sample data. The performance and accuracy measurements are described in the next two subsections.

System Performance

The overall system performance from a functional perspective was determined by observing the computer system clock and calculating the elapsed time of the deception detection prototype system operation from program launch to completion of critical BDI agent centric operations within the system. System performance measures were calculated using elapsed time between the start of BDI agent related tasks and their completion. The prototype performance tests included measuring the time needed to complete the following tasks:

1. Creation of the input vector CSV file (“vectors.csv”)
2. Backup of the “vectors.csv” file

3. Backup of the “weblogeentry” database table
4. Deletion of the “vectors.csv” file in preparation for another program run.

The times collected were rounded to the nearest second as the system performance was measured with respect to human task completion abilities. Human task completion standards are typically measured in seconds.

The most important performance test of the deception detection prototype system was the testing of the processing NN input vectors and placement into the “vectors.csv” file. This was chosen as a metric because it is the primary automated function of the system which was designed to run without human interaction using autonomous BDI agents. It is the most complex and computing intensive. The BDI agent modules must access the database using the JDBC API using prepared statements and execute queries, then process the results line by line through 10 input vector generation algorithms, update the “vectors” table and perform file operations. All results were determined rounding to the nearest second. The results of ten test runs indicate that the BDI agents successfully performed the following steps within 14 seconds on average

1. The program uses the semaphore to grab the processing thread. Then the BDI agent accesses the database using a JDBC API prepared statement and removes all content from the “vectors” database table of the Roller Weblog server “rollerdb” database, closes the JDBC connection to the database.
2. Once again the program accesses the database using JDBC API and prepared statements.
3. The program extracts data from the “weblogentry” database table using a query iterate through all query results one by one.

4. The program processes the results query output line by line through 10 input vector generation algorithms.
5. The program executes another prepared statement to update the “vectors” database table by placing the results of the algorithms into the table
6. The program executes a prepared statement which queries the “vectors” database table
7. The program creates the “vectors.csv” file
8. The program enters the data from the query of the “vectors” data base table into “vectors.csv” file line by line.
9. The program closes the JDBC database connection.
10. The program releases the processing thread
11. The program sends a BDIMessageEvent indicating task completion and thread availability.

The results of the “vectors.csv” file creation test are shown in Table 5 of Appendix B. This portion of the system is the core of the prototype due to its complexity and the need for BDI agent coordinated activities, autonomy and synchronization. This was an important metric because the system is using BDI agents to perform autonomous and automated file system tasks that are vital to operations. The BDI agents use the semaphore and inter-agent communications to synchronize the various file operations as described in the system operation section of Chapter 3. As discussed the system relies on a BDI agent software based FSM. The test results of the creation of the “vectors.csv” file were collected by comparing the prototype start time to the time that the creation of the “vectors.csv” file was complete.

Additional performance measurements in the form of file system and RDBMS management autonomous activities were taken. This included the creation of "vectors.csv" backup file which was determined by taking the time difference between the creation of the "vectors.csv" and the time that the backup "vectors.csv" file was created. Tests were conducted ten times to ensure that the system was in a stable state and to be able to obtain an average of the test times. During the ten test iterations the backup "vectors.csv" file was created within one second. The results of the ten test runs are contained in Table 6 of Appendix B.

Another performance metric was collected concerning the backup of the "weblogentry" database table. This was obtained by calculating the elapsed time between the start of the deception detection software and the time of the creation of the database backup. The importance of this metric is linked to COOP and reduction of risk to the Weblog server environment and its underlying data. It can be an implication of the type of risk management operations that BDI agent autonomous behavior can offer. COOP related tasks are an important indicator of whether automated BDI agent risk management might offer autonomous behavior, speed and accuracy for such tasks. The results showed that on average the "weblogentry" database table was automatically backed up within 1 second from the start of the deception detection system prototype by the BDI agent modules. The results of ten test runs are contained in Table 4 of Appendix B.

The final BDI agent based task performance measurement involved determining the elapsed time between the deception detection system prototype start and the deletion of the "vectors.csv" file. The "vectors.csv" file must be deleted for the system to run

again. The BDI agents had previously created the backup “vectors.csv” file. The “vectors.csv” file deletion process was completed on average within one second after the backup of the “vectors.csv” file creation during ten test runs. Table 7 of Appendix B contains the results of the test runs. This test was important in that it determined that the system could perform BDI agent based autonomous file system operations effectively and rapidly.

Neural Network Performance

For validating the effectiveness of the input vectors a validation methodology was developed as described. The data from the NN function was obtained from classification runs to obtain the best training and validation sessions. Using randomization and normalization and an RBF kernel function, the NN module attained 81.8% accuracy with 162 samples. The breakdown of the samples in the training and validation sample set was Table 2.

Classification validation results.

Classification Accuracy Calculations during Validation			
Correctly Classified Instance Rate	81.8%	Incorrectly Classified Instance Rate	18.2%
True Positive Rate “deceptive”	77.8%	False Positive Rate “deceptive”	22.2%
True Positive Rate “not deceptive”	86.7%	False Positive Rate “not deceptive”	13.3%

81 samples “not deceptive” or True and 81 samples “deceptive” or False. Table 2 shows the validation results. Normalization in this context means that the data was transformed into values between zero and one for optimum NN performance (Whitten, Eibe & Hall, 2011). Randomization in this context means that that order of the entries in the CSV file were placed in random order to ensure that random sampling is assured as the sample set is presented to the NN for training and validation. An 80-20 percent training to validation scheme was used. This means that 80% of the 162 samples were used to train the NN and

a randomly selected 20% of the 162 data samples were used for validation. After validation the training algorithm produced by the NN was discarded before moving into the testing phase.

For testing, a larger data set was introduced. To ensure that the system did not overlearn the data the WEKA NN software was loaded on a separate test computer. This served to allow the testing to be conducted using the same randomization, normalization and RBF kernel function without the possibility that the system had prior exposure to input vector data samples. The NN interface and prescribed process for dividing the data sample file into two separate files was employed to ensure that testing samples were not examined by the NN during training. The instructions for dividing the samples into a training file and a testing file are delineated in Appendix C. The amount of input vector samples numbered 212. The breakdown of the input vector sample set was 106 “not

Table 3.

Classification test results.

Classification Accuracy Calculations During Testing			
Correctly Classified Instance Rate	85.7%	Incorrectly Classified Instance Rate	14.3%
True Positive Rate “deceptive”	80%	False Positive Rate “deceptive”	20%
True Positive Rate “not deceptive”	90%	False Positive Rate “not deceptive”	10%

deceptive” and 106 “deceptive”. According to Whitten et al. (2011) dividing the samples into training and testing files using this method ensures that the NN does not over-learn the data as the test data sample set is only accessible during testing. To ensure that the NN only classified the intended target values of “deceptive” or “not deceptive”. Only nominative data in the “TorF” column of the input vector sample files was used. A screen capture of the WEKA GUI in Figure 4 shows evidence of classification target restriction to only classify the nominative values in the “TorF” column. Figure 4 shows that on the

classify tab of the NN interface that “nom TorF” is selected in the mid left hand side of the user interface just above the start button. The shorthand for “not deceptive” is “T” and the shorthand for “deceptive” is “F”. The results of the testing using 170 samples for training and 42 samples for testing were 85.7% accuracy. Table 3 shows the classification testing results. The raw file containing the 212 samples, the training sample file and the testing sample file were converted to “.arff” file format also known as ARFF format. ARFF format is a special WEKA format that allows operations using the WEKA NN tool set beyond what can be accomplished using CSV file format (Whitten, Eibe & Hall, 2011). The training sample file contents are contained in Appendix F. The testing sample file contents are contained in Appendix G.

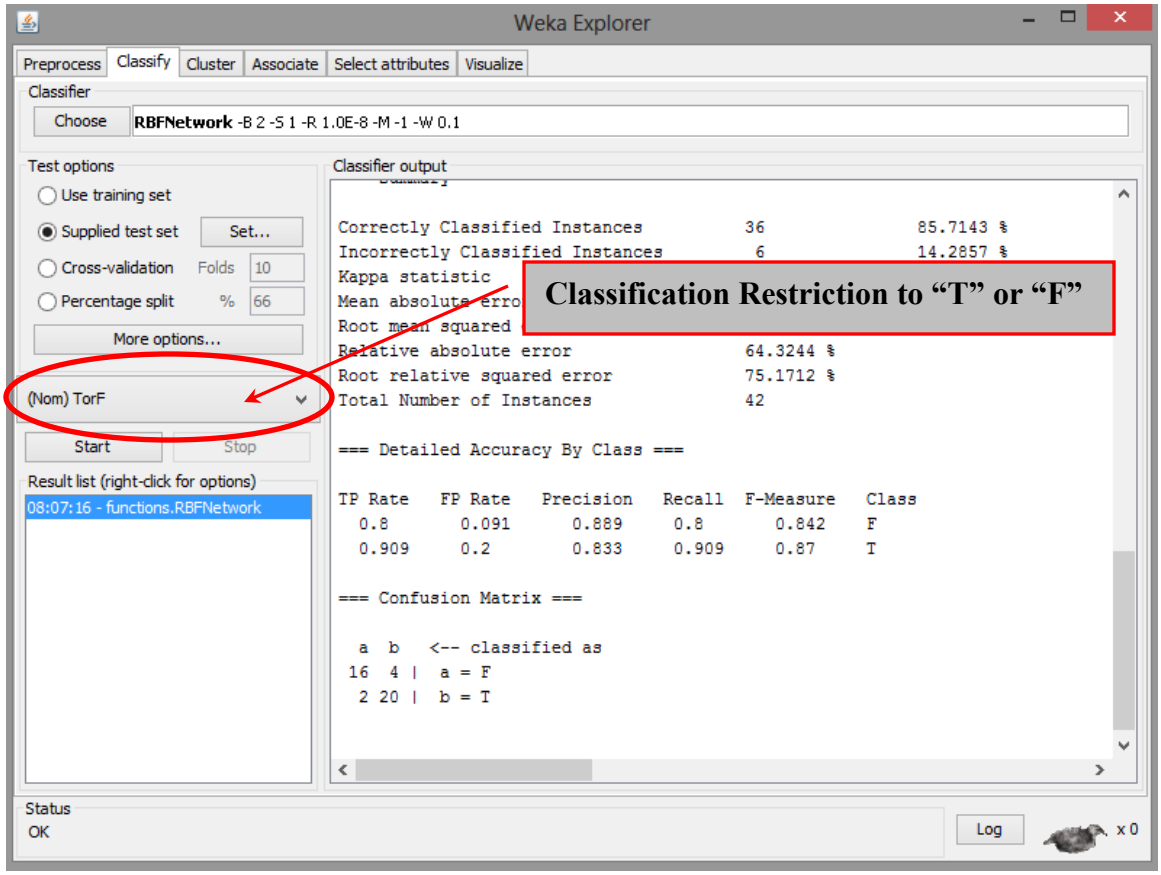

Figure 4. WEKA TorF (“not deception” or “deception” classification restriction).

Accuracy was derived using the Accuracy formula depicted in Figure 3. The True Positives (”deceptive” or “F”) findings were added to the True Negatives (”not deceptive” or “T”) findings. The sum of these were divided by the amount of input vector samples attempted. For this test the NN found 16 of the samples to be “deceptive” or “F” and 20 of the samples to be ”not deceptive” or “T”. The total number of samples which the ARTMAP NN attempted to classify contained in the input vector sample test file was 42. The False Discovery Rate as described in Figure 3 was 20%. This was derived by dividing the amount of False Positives (”deceptive” or “F”, but not properly classified) by the sum of False Positives added to True Positives (”deceptive” or “F”, properly classified). This means that the system classified deceptive Weblog entries correctly 80% of the time. Although this percentage seems low for False Discovery of deceptive entries, if we examine the False Negatives using a similar approach by dividing the number of False Negative discoveries (”not deceptive” or “T”, but not properly classified) by the sum of False Negatives plus True Negatives (”not deceptive” or “T”, properly classified), then the percentage is a much lower False Discovery rate (10%) for “not deceptive” entries. Figure 5 shows this formula for clarification.

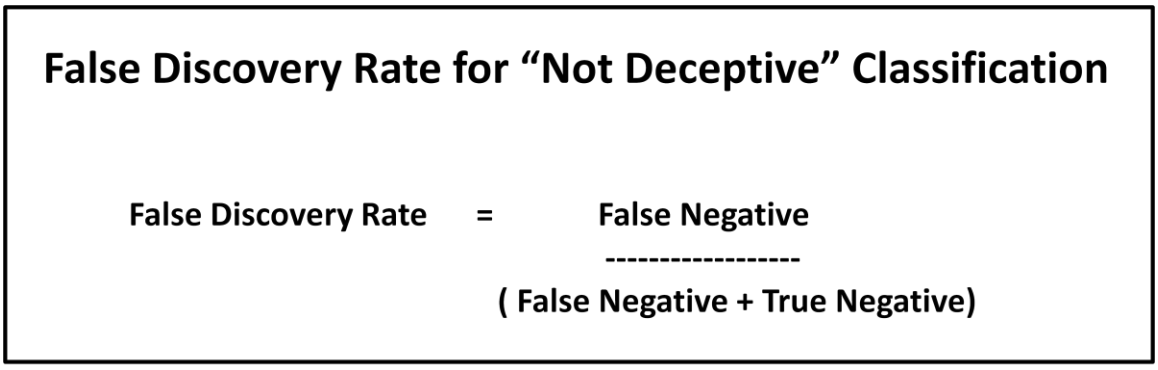

Figure 5. “Not Deceptive” False Discovery Rate Formula.

This means that the system was more effective at determining if a Weblog entry was truthful than it was determining if an entry was deceptive. In other words the system classified truthful Weblog entries correctly 90% of the time. This is quite useful as the classification scheme is simple binary and the knowledge that only 10% misclassified True Negatives could assist in further identification of deceptive entries using additional analysis and tools as a future research effort.

Findings

This research effort demonstrated that a successful deception detection prototype system development is possible. The system developed proved that a system could analyze text in online data communications systems to extract linguistic cues to deception with automation and autonomy. The system used BDI agents to provide automation to extract input vectors by using algorithms developed using the Java programming language. The system performed well processing 212 Weblog server records for entry into the NN within seconds. The prototype performed more rapidly than a human could process this amount of records as each record was one paragraph in length. The input vectors were created by a series of algorithms designed to extract linguistic cues to deception (Toma & Hancock 2010). The input vectors created were sufficient for processing by the NN with 85.7% accuracy in determining whether a deception had occurred or not. This is quite positive since a human can only discern the difference between the truth and a deception about 50% of the time in any media (Zhou & Zhang, 2008).

The design of the deception detection prototype system is modular and flexible enough to be adapted to a real-world data communications system. This is because of the

use of standard Java objects and the JDBC API which allows the system to operate on many RDBMSs. The BDI agents use standard Java based objects to accomplish automation and perform tasks autonomously in a synchronized manner using a BDI agent software based FSM (Miles & Tashakkori, 2010). The FSM allows the BDI agents to synchronize their automated activities. This was a critical design factor for interaction with the system back-end database. If the system did not possess the BDI agent software based FSM, synchronization would need to be accomplished manually writing Java methods for thread locking inserted within the source code (Agent Oriented Software, 2011). Writing Java thread locking methods into the code base would add complexity to the system design, add hours of programming to the code creation time. Manual coding would also add the potential for the system subcomponents to get out of synchronization and potentially cause the system to malfunction.

The NN selected was an excellent fit for the deception detection prototype system design. The approach used was to train the NN using the input vectors generated by the BDI agents, validate the training and finally to train and test using separate training and testing files. For the purpose of test a separate computer configured with the WEKA NN software was used to run the testing portion. This was done to prevent any possibility of NN over familiarization with the input vector samples. The input vectors output by the BDI agent based input vector generation algorithms were of sufficient quality that only randomization and normalization were used in preprocessing them. This is very encouraging considering that the 85.7% overall accuracy rate was much higher than the deception detection accuracy rate expected to be achieved by humans, which is about 50% (Zhou & Zhang, 2008). Due to the modular design of the system developed and the

portability of the modules many NN solutions or other Machine Learning modules could potential be integrated. Although the final build of the prototype integrated a WEKA NN, during development of the system, other tools were successfully integrated as subcomponents for trial. These included:

1. DTREG- A comprehensive commercial Machine Learning tool set.
2. ENCOG3-An open source NN tool.
3. MATLAB Neural Network Toolbox- A commercial NN add on to the MATLAB statistical research tool set.

The implication and importance of being able to integrate differing NN software suites as modules is that further research can be conducted by exploring many Machine Learning classification techniques in efforts to increase deception detection accuracy by easily swapping them.

Summary of Results

The online deception detection prototype system resulting from this research successfully proved that combining BDI agent based AI with NN based Machine Learning AI is an effective technical approach to deception detection capability development with Information Assurance implications (Gondotra, Singhal & Bedi, 2011). The overall performance data gathered during testing proved that BDI agents can autonomously perform deception tasks in online data communications systems rapidly, with much better accuracy than humans are able to do. The deception detection prototype worked through 212 paragraphs of potentially deceptive text and created usable input vectors for the NN within seconds. The BDI agent software based FSM handled the synchronization and execution of the BDI agent tasks by controlling access to the

processing thread (Miles & Tashakkori, 2010). This included queries to the back-end RDBMS, table updates, file management operations and execution of the main program containing the ten input vector generation algorithms. The BDI agent software based FSM gave the prototype stability and testability (Telang, Meneguzzi, & Singh, 2013). Humans can only successfully detect deception without the aid of computers about 50% of the time (Zhou & Zhang, 2008). The deception detection prototype developed for this research successfully classified entries as "deceptive" or "not deceptive" 85.7% of the time. Although this rate is not without need for improvement, it shows a 35.7% increase in accuracy over human efforts to detect deception and is quite an improvement. The False Discovery rate for deceptive online data communications entries was 20% while the False Discovery rate for truthful entries was a mere 10%. Although the system was designed to detect deceptive entries during conceptualization, the data shows that it detected the true entries 90% of the time versus 80% of the time for deceptive entries based on calculating of the respective False Discovery rates. This is still quite positive because 90% of the truthful entries can be eliminated as well as 80% of the deceptive entries from the data corpus after initial classification. The 10% of the remaining truthful entries and the 20% remaining deceptive entries could be further analyzed with other techniques such as fuzzy c-Means classification capabilities developed during future research efforts (Ross, 2011).

The deception detection prototype system was built using the Java programming language. The JDBC database connectivity API was incorporated. The JDBC database connectivity API specification gives the system flexibility to be adapted to many online data communications systems with a backend RDBMS. The system could be readily

integrated into a real-world online data communications system where text has been previously entered and a need to analyze whether the text contained in the database possesses deceptive characteristics. This could apply to social media data communications systems, online marketplace systems, internal business data communications systems and many other specialized applications where text based data can be analyzed. The modularity of design and incorporation of the BDI agent software based FSM allows for rapid integration, stability of design and testability once in place (Miles & Tashakkori, 2010).

The modular design of the deception detection system prototype also allows for flexibility in the selection of Machine Learning capability incorporation. This means that the NN implementation could be swapped out for DTREG, ENCOG3, MATLAB Neural Network Toolbox or many other commercial or open source NN software subsystems. What's more, a custom designed NN could be developed specifically for deception detection classification purposes and integrated into the prototype design as part of a future NN research effort.

Chapter 5

Conclusions, Implications, Recommendations and Summary

Conclusions

This research has combined Artificial Intelligence (AI) methods to achieve objectives in Information Assurance research. The approach used intelligent agents in the form of BDI agents and Machine Learning in the form of NN classification capabilities operating within an online data communications environment. Employing a combined AI approach in a development effort was the overall design consideration for the creation of a deception detection prototype (Gondotra, Singhal & Bedi, 2011). A BDI agent software based FSM afforded the deception detection prototype system stability and consistency in operations while synchronizing activities during execution (Miles & Tashakkori, 2010). A NN was integrated into the system for the classification of the input vectors created by the input vector generation algorithms which were developed during design.

During the validation portion of NN testing, the NN achieved 81% accuracy using an 80-20 percentage split between training and validation sample sets. The sample set used for the validation phase contained 162 samples prior to the 80-20 validation split. The output of the validation portion of NN classification is contained in Appendix D. For testing, additional data was gathered. The testing sample set contained a total of 212 samples. After the validation phase another dedicated test computer physically separated from the machine used for validation was configured with the NN software. This setup was devised to ensure that the NN did not become over familiar with the input vector data sample set. This was assured because the test machine's NN software had not performed

any NN training on the input vector sample sets prior to the test phase. During the testing phase the samples were separated into two separate files to ensure that during training of the NN there was no access to the test input vector sample file. The file split of the training sample and test file was an 80-20 percent split to mirror the validation hold-back split conducted previously. During the testing phase the training input vector sample file contained 170 samples. The test input vector sample file contained 42 samples. Only randomization and normalization preprocessing was performed on the input vector samples prior to NN training. During testing the NN achieved 85.7% accuracy overall in classifying the 42 test input vector samples based on the training of 170 training input vector sample sets. The output of the NN from the test runs are contained in Appendix E. This accuracy rate was sufficient to determine that the BDI agent based online deception detection system prototype developed was a success. The system autonomously generated usable input vectors from an online data communications system which were usable for the NN to perform deception detection classification. The results were also a proof of concept that the system could perform many of the critical deception detection operations autonomously and with performance that displayed stability, testability and very rapid processing. The research effort also proved that the system could be integrated into an operational online data communications environment with little impact and ease of integration.

As previously stated, the research goal is: To create capabilities which autonomously and automatically detect deception in online data communications systems. The capabilities developed will detect online data communications deception to

realize this goal. To measure accuracy, evidence of a deception will be uncovered in text entered by the user.

The prototype developed detected deception in online data communications and met this goal. To measure accuracy, evidence of a deception was uncovered in text entered by users. The objectives were accomplished using a developmental research methodology resulting in a prototype that effectively acted as a proof of concept for a BDI agent based online deception detection system prototype with an integrated NN classification capability. The BDI agents provided automatic and autonomous behavior for essential system functions. System processing thread synchronization was achieved using a BDI agent software based FSM. Classification was accomplished using a NN that provided over a 30% increase over human ability to accurately classify online data communications as either "deceptive" or "not deceptive". The system performed all associated tasks rapidly as previously defined for the purposes of this study. The performance tests and the NN classification effectiveness evaluations support the accomplishment of the research goal and are contained in the Appendices.

Strengths

The BDI agent software based FSM proved highly testable, stable, scalable and adaptable operating in the deception detection prototype environment (Miles & Tashakkori, 2010). This is a major strength of the deception detection prototype design as the integration of capability modules that require synchronization and control of the processing thread could be integrated very easily. The stability of the BDI agent software based FSM facilitated the testing of newly integrated modules. BDI agent software based FSMs have proven scalable in the gaming virtual environments employing them (Telang,

Meneguzzi, & Singh, 2013). Scalability of the FSM will be advantageous for future research moving toward large online data communications data containers. These qualities will offer adaptability for future research and for practical industry usage alike.

The modular design of the Java based components comprising the system will allow for system portability and facilitate its use in many environments. The deception detection prototype system design will allow for variations in database types using JDBC API for many commonly found RDBMSs. Modification of the input vector generation algorithms to match new data sets would not be difficult due to their modular and practical design. These algorithms can be recoded and recompiled and easily integrated into the BDI agent based prototype due to the Java programming language portability features. Java archive (JAR) files were used to ensure portability of the source code. The JACK BDI agent development environment was compatible with the Java programming language and its associated class libraries and APIs such as the JDBC database connection API facilitating portability of executable code. This is another major strength.

Weaknesses

A major weakness of the research is that NNs are black boxes and there is no real way to know how the hidden layer worked to come up with a training algorithm (Heaton, 2011). This is especially true when using Machine Learning software packages as add on modules to research prototypes. Although the NN software module chosen for this implementation is well documented and the source code is available, all NNs contain at least one hidden layer of neurons. Therefore it is never really clear how the Machine Learning process is occurring or exactly how the NN arrives at its classification

decisions. However, the output is reliable and has a solid foundation as proven by prior research.

Another weakness is the small data set used. The data was collected specifically for the research conducted and was limited to a single topic. A much wider scoped data set with many more samples would have been more desirable. However, time constraints and the need to do a proof of concept for the prototype were factors in using a small data set. Assurance of data fidelity was the primary driver behind the data collection methodology employed.

Limitations

A limitation for the research effort existed in that data was generated instead of gaining access to a large data source. The process of data collection was time consuming as data was entered into a Weblog server browser based interface instead of working with a preexisting corpus of data. This limited the study in the amount of data that could be obtained in a reasonable amount of time. It was chosen as a proof of concept of end to end data entry into an operational online data communications environment. The method was also selected as the data entry interface for data collection realism.

Implications

The research conducted as described in this research and the BDI agent based deception detection prototype which was developed as a result, extends the body of research which explores the combined AI approach to IA automation employing intelligent agents in concert with a NN. The ability to combine automation with Machine Learning techniques also contributes to professional practice if the research is adapted to real-world problems such as eliminating pre-existing deception in online data

communications. If adopted, a deception detection module integration into existing online data communications systems could potentially reduce online fraud attempts and constrain the ability of individuals to commit identity theft crimes. This assumption is based on the fact that most of these crimes are initiated using seemingly innocent deceptions in online data communications (Boongoen & Shen, 2009).

The use of the BDI agent software based FSM, although not new technology, adds stability to the system which may be suitable to adaptation for very large data sets and real-world implementations. These real-world adaptations could include social networking sites and online marketplaces. The technology developed could be adapted due to its modularity, and portability afforded by the Java development language used and the use of the JDBC API. This allows for the capability to be employed in almost any online data communication environment containing a backend RDBMS. The design allows for the inclusion of a wide variety of Machine Learning tools integrated as subcomponents. The modular design will also afford further research as new capabilities that can be brought to bear to increase the accuracy rate of the deception detection. This research could be adapted to the DoD Active Cyber Defense Strategy. Active Cyber Defense is defined by the U.S. Government as follows:

> Active cyber defense is DoD's synchronized, real-time capability to discover, detect, analyze, and mitigate threats and vulnerabilities. It builds on traditional approaches to defending DoD networks and systems, supplementing best practices with new operating concepts. It operates at network speed by using sensors, software, and intelligence to detect and stop malicious activity before it can affect DoD networks and systems. As intrusions may not always be stopped at

> the network boundary, DoD will continue to operate and improve upon its advanced sensors to detect, discover, map, and mitigate malicious activity on DoD networks (U.S. Department of Defense, 2011, p. 7).

The deception detection system prototype could potentially be considered a new operating practice used to detect and stop malicious activity early as part of the discovery process.

Deception detection in online data communications techniques could be integrated as part of a dynamic and multi-faceted IA future research effort incorporating AI in many forms to be able to detect and deter malicious activity. This could range in capability from extending GP research for automatic policy enforcement using combined discovery of potential malicious activity beginning with deceptive communication (Lim, Chen, Rohatgi, & Clark, 2009). The multilayer defense in depth approaches using AI research combining Fuzzy Logic, intelligent agents and NNs are designed to harden systems against compromise and increase discoverability of malicious activity early in a cyber attack cycle (Gondotra, Singhal & Bedi, 2011). This hardening can be extended by adding a prevention dimension with a deception detection capability addition. For boosting the accuracy and adding an adaptive characteristic to the prototype, the use of Fuzzy Logic classification to determine the degree to which the entries are included in the classification sets of "deceptive" or "not deceptive" could be undertaken. Ross (2011) describes the Fuzzy c-Means fuzzy set methods for improving classification accuracy and dealing with unclear and ambiguous data sets. Fuzzy c-Means classification could also prove useful as an added module to prevent online communicators from modifying behavior to attempt to fool the system.

Recommendations

Recommendations for future research include a large system integration of the deception detection system prototype modules into a deployed real-world online data communications system. A social media site or online marketplace might be suitable. This would allow for a much larger data container to be examined. The larger data set would contain more diverse topics for the deception detection prototype to classify, giving the prototype more diversity in data classification. The NN training sets would be larger. The larger NN training sets could potentially boost the accuracy rates of the deception detection NN discoveries (Boongoen & Shen, 2009). The more diverse data topics would allow for modification of the input vector generation algorithms. The configuration of which could be retained specifically for a given topic or data set. This is important because the persistent nature of online data communications and the commonality of interest of persons adding to the corpus of data would assure that re-use and retention of the input vector generation algorithms would be valuable in practical usage.

Another recommendation is the exploration of other Machine Learning techniques for inclusion in the research. This may show promise for increased classification accuracy in the deception detection prototype. These could be easily integrated into the prototype given its modular design. This may prove an essential research area as other online data communications systems and their data may be better suited for training and testing using various Machine Learning techniques. For example, some potentially useful Machine Learning capabilities worthy of further research efforts might include Tree Boost and Support Vector Machine (SVM). TreeBoost, which is also

known as stochastic gradient boosting, is not a NN but is promising because it increases the accuracy of predictive tree algorithms by repeated application of the algorithms and combining outputs with weights to the data in an effort to reduce error rates. TreeBoost usually achieves better accuracy rates than single tree decision models (Sherrod, 2013). TreeBoost is similar to decision tree forest Machine Learning, however instead of parallel decision tree growth, TreeBoost decision trees are grown in series. The algorithms attempt to reduce error with each tree in the series. TreeBoost is included in the DTREG Machine Learning software package which was successfully integrated into the prototype and evaluated but not selected.

In comparison, SVMs divide data into two categories for classification using an N-dimensional hyper-plane. The SVM algorithm finds an optimal algorithm to separate clusters on either side of the hyper-plane. The SVM may be well suited for two target classification research such as deception detection in online data communications. The dimensions of the SVM used for hyper-plane division are based on the amount of predictor variables, so that N would be equal to the number of predictor variables which are also known as input vectors. SVMs are similar to NNs in construction (Sherrod, 2013). SVMs are included in most Machine Learning software distributions.

Another recommendation is future research for the introduction of Fuzzy Logic into the online deception detection system prototype. This would allow for further analysis of misclassified data samples and could be added as an additional layer of processing to increase overall classification accuracy. This would allow the system to have an adaptability feature as human behavior may change the way deception occurs over time. This can be expected as people discover that deception detection techniques

are being used. People will undoubtedly modify their deceptive behavior requiring adaptation by the deception detection system (Gondotra, Singhal & Bedi, 2011).

Another recommendation is to include policy for enforcement of the requirement for truthful data communications as part of conditions of Information System resource usage and include enforcement measures and consequences. It could be combined with a user acknowledgement that there is an integrity checking capability installed. This could be a disclaimer and a condition for using the information system, or it could be built in to justification for data access. For example, a worker at a company seeks access to the Human Resources databases which contain PII, salary and disciplinary data. Before being granted access the individual must write a justification in free form paragraph format giving all of the facts behind the request. These facts would include who authorized the individual to seek access, why the access is needed, what particular data is needed, what will be done with the data, where it will be stored, data access times and how long the data will be retained. The system could check the justification for deception using a similar approach as is presented in this report in an attempt to eliminate insider data theft, or other malicious activity. This capability could be combined with Genetic Programming techniques that allow for automatic and dynamic policy enforcement as described by Lim, Cheng, Rohatgi & Clark (2009). Such an approach would allow for dynamic policy generation linked to risk assessments which are based on the deception detection system.

Summary

As previously stated, the goal of this research was achieved. How it was achieved from an end to end perspective is described in this summary. A developmental research methodology was selected to develop an online deception detection prototype. An

operational environment was needed to integrate the prototype into. This consideration gave the prototype development effort a real-world operational pertinence. The operational environment chosen was Weblogging. The Apache Roller Weblog server was selected due to its open source availability and ease of integration. Roller required MySQL server as the RDBMS backend and Apache Tomcat as a Web Servlet container to allow web access to the Roller Weblog user interface and administration suite (Apache Roller, 2011). The environment was rapidly configured in compliance with the Roller, MySQL and Apache Tomcat distribution documentation with no variations. The selection of this environment was based on the knowledge that Apache Roller is a Java based web online data communications system with a backend RDBMS. This environment is a good representation of the online data communications systems which are prevalent on the Internet.

Once the Roller Weblog server was fully implemented and tested for functionality development of Java programming language modules began. The modules used the JDBC API driver to interact with the MySQL RDBMS behind the Weblog server. This was the starting point for much of the activity. Once database access was achieved using the JDBC API driver specification the Java modules were developed. There were able to successfully execute data manipulation and table creation, database backup, table updates and queries. Once this capability was tested for functionality modules were created to extract user Weblog entries from the backend database. The entries were inserted into Java String type variables and processed through ten input vector generation algorithms to assign numerical values to the characteristics of the data found within the text entry. These values were placed into a table created specifically to hold the values and maintain

the fidelity of the data and to be able to trace the Weblog entry creator name back to the original entry. The processing of the Weblog server entries was conducted line by line as the system was queried using the JDBC API and using Java methods that allow query result sets to be iteratively accessed and processed. These types of interaction with the database were accomplished using the Prepared Statement Java method when it was necessary to have modules conduct database operations. As more and more functionality was incorporated into the modules it was necessary to start building the BDI agent based autonomous functionality. The Java modules were integrated into the JACK BDI agent based development environment. When possible the Java modules were transformed into JAR file format for portability. These JAR files allow for all of the library dependencies to be included into a single executable file. The BDI agent capability portion was used to allow for the autonomous interaction with the MySQL backend RDBMS and processing of query results.

A BDI agent software based FSM was created which allowed the agents to synchronize RDBMS access and other activities. This was important due to the interaction with the database and the need to synchronize activities (Agent Oriented Software, 2011). If activities in an execution environment involving database interaction get out of synchronization, the reliability and stability of the program suffer. Once the Java modules were integrated into the BDI agent development environment, functional testing was conducted. After successful integration of all the modules developed using Java, other functions were developed for BDI agent execution. This included the launching of MS Excel and the NN interface after each processing run as well as opening the file containing the input vectors for examination and processing. The prototype

displayed stability throughout the development. Integration from the NetBeans development environment to the JACK BDI agent IDE was greatly facilitated using the JAR files. JAR file portability allowed for rapid integration from development as a Java module to a capability that a BDI agent could execute as part of its plan library.

Data collection was conducted using volunteers to provide a truthful description of their house. Volunteers were not informed of the purpose of the data collection. After users provided a truthful description they were asked to provide a deceptive or fabricated description of their house. The volunteers numbered 106. There were 212 paragraphs entered into the database using the Roller Weblog server user interface. Each data provider was given two user accounts; one account was used for truthful Weblog entry creation and the second for a deceptive Weblog creation. This was useful for training the NN as it is necessary to provide the "deceptive" or "not deceptive" classification during the NN training phase. Preliminary testing during development verified obtaining consistent results for classification accuracy required at least 150 data samples for classification for training and validation using the NN. This was based on examination of input vector data samples provided with NN software packages that were evaluated for potential inclusion in the prototype (Whitten, Eibe & Hall, 2011).

System performance testing was conducted after data sample inclusion in the Roller Weblog server RDBMS was complete to get an idea how well the system could perform database and other operations. The system was very stable and no system crashes occurred. This was attributed to the use of the BDI agent software based FSM approach adopted during development. The system did not get out of synchronization and the performance was nearly identical through all of the performance test runs. The system

produced usable input vectors rapidly from raw data within seconds of the start of the program run. The execution times of the various modules executing in the deception detection system prototype are contained in Appendix B.

Once a certain degree of functionality was built into the deception detection system prototype, a NN was selected for integration as a module. The selection of the NN was chosen because of its Java based design, open source availability, ease of use, availability of documentation and availability of source code (Whitten, Eibe & Hall, 2011). The NN was integrated by taking its executable JAR file and having the BDI agents launch it after the input vectors were generated and all file and database operations were complete. The NN interface was used to randomize and normalize the data during a preprocessing stage. A validation run was conducted by splitting the data into 80% of the samples for training and a 20% holdback for validation. Once the system achieved performance of over 80% it was determined to be a successful validation run. The validation phase input vector data samples numbered 162. To test the accuracy, more data was gathered using the same instructions and method as the prior data collection effort. After gathering 212 total samples a test plan was developed. For the test, data was not held back for validation during NN training. Instead the data was randomized and normalized during preprocessing. An additional step was included to populate the samples into two separate files. The file containing the training set contained 80% of the samples which amounted to 170 individual samples. The file containing the test set contained 20% of the samples or 42 individual samples. The files containing the input vector samples were tested on a separate dedicated computer to ensure that the NN was

not over familiar with the data. The system achieved 85% accuracy implementing this test methodology.

One implication of this research is that in online data communications, users may not be able to enter deceptive information without being discovered. This is important when considering that many identity theft and other fraudulent crimes are initiated using a seemingly unimportant online data communications deception (Boongoen & Shen, 2009). Uncovering deception in online communications could have implications in legal matters as well as for inclusion in the Department of Defense's Active Cyber Defense Initiative (U.S. Department of Defense, 2011). This research acts a proof of concept that using AI for IA tasks as part of an Information System prototype development effort can be extended to the area of deception detection in online data communications. The prototype created using a developmental research approach was effective in classifying online data communications entries in a Weblog server with greater accuracy than human ability. The speed at which the system performed automated and autonomous functions was very rapid given that 212 paragraphs of data were analyzed and processed into input vectors in seconds. BDI agents were a critical design feature due to their ability to provide a software based FSM to allow synchronization of program thread activities. This allowed the system to be very stable. Although a BDI agent software based FSM is commonly used in the video game software development industry, it proved well suited for inclusion within the deception detection prototype development research effort. Without the FSM capability, much more coding would have been required to ensure thread locking as the modules executed. The FSM ensured that synchronization of the modules was maintained.

The inclusion of the NN was the centerpiece of classification and worked well with the input vector data generated by the system. The NN provided classifications for "deceptive" or "not deceptive" categories in a 2 class classification scheme achieving 85% accuracy. No complex preprocessing schemes were needed to boost the accuracy rates. The only preprocessing conducted on the data was a randomization and a normalization which are standard procedures for input vector treatment prior to NN training and testing (Sherrod, 2013). The use of Java based programming techniques, methods and packages and in particular the JDBC API allows for the deception detection prototype to be adaptable to almost any system which includes a backend RDBMS. The JDBC API specification allows Java based programs to connect and run prepared statements against almost any database. These include Oracle, Sybase, DB2 and others. This adaptability of the prototype will allow future uses of the prototype. Of particular interest might be to allow the attachment of the prototype to much larger pre-existing data sets for additional research or industry practice purposes.

Although the corpus of data was small for this proof of concept effort, enough data with high fidelity was collected to verify that the system could analyze and process free form text and produce usable input vector data. This approach of gathering first hand data, although limited, was very effective for the proof of concept. The gathering of first hand data was very time consuming as each user individually provided two paragraphs of information. However, the fidelity of the data was deemed to be much higher than if an online survey tool or other means were used where there was less control over the data collection process. Future research could involve a larger corpus of data which is pre-existing. A larger corpus of data will most likely allow for better training algorithm

development within the NN training phase and higher classification rates. A larger pre-existing data set also saves time and allows more time to be devoted to other areas of future deception detection research. Using pre-existing data samples for NN training and testing is also quite common in other research efforts (Sherrod, 2013).

The development of the deception detection prototype could easily be adapted to business practice to solve the real-world problem of discovery of deception in data communications. Eliminating the ability to lie in online data communications could prove to be a huge step forward in reduction of fraud, identity theft and other crimes and malicious activities which originate on the Internet (Boongoen & Shen, 2009). The deception detection system could be valuable for research, industry or Government usage. In particular the capability could be part of the U.S. DoD's Active Cyber Defense Strategy. The intent of the DoD's Active Defense Cyber Strategy is use of new operating concepts to detect and stop malicious activity early in the lifecycle (U.S. Department of Defense, 2011).This gives the research a wide scope of implications for usage and research in academia and practice.

A dynamic multi-dimensional approach to IA could be developed with deception detection as a subcomponent. This could be part of a large scale future research effort using multiple AI based IA capabilities to detect and deter many forms of malicious activity. Such an effort could extend automatic policy enforcement using GP research as described by Lim, Chen, Rohatgi, & Clark (2009). Using deception detection as part of policy enforcement could help to ensure that users of Information Systems conform to truthful data communications behavior standards or risk consequences as delineated in the policy. Another implication for implementation of the deception detection prototype

includes using the prototype as a component of a multilayer defense in depth approach using AI capabilities combining Fuzzy Logic, intelligent agents and NNs to harden systems (Gondotra, Singhal & Bedi, 2011).

Additionally, research into other Machine Learning techniques for integration into the deception detection prototype may prove that other methods are more effective for a diverse set of online data communications systems. Other Machine Learning modules can be easily integrated into the prototype much as the WEKA NN module has been. Testing of these Machine Learning techniques could be a rich area of research resulting in customized deception detection algorithm development.

Appendix A

Weblog Data Entry Instructions

Data Collection 1

Introduction: This research deals with online data communications. The participants will enter data into a Weblog server. The blog server is self contained on the data collection laptop machine as a local web server and is not posted on the Internet. The identities of the participants will not be revealed and the data entered into the Weblog server will not be used. Instead metadata about the Weblog server entries will be used in the research. Please be as honest as possible in your responses, use complete sentences, punctuation and use the spell checker/word replacement function where appropriate as you enter data.

Instructions: In the Weblog entry space provided describe your house please. Below are some questions and prompts to help you think about some aspects of your house's description. Please be as honest as possible:

1. Can you describe the color scheme?
2. Describe the shingles or siding materials.
3. Describe the windows, are there many windows? For example are there picture windows installed or are they multi paned windows?
4. Please describe the landscaping, for example are there lots of trees, grass and shrubs?
5. Does your home have a backyard, please provide a description.
6. If there are trees what types of trees are on the property?
7. Please describe the interior of the house, what are the rooms like?
8. Please describe the style, for example Colonial, Split Level, Victorian, etc. Are there many rooms? Please describe the rooms.
9. Describe your furniture please. For example what is the upholstery like in your living room?
10. What types of window coverings do you have? Please describe them.
11. How would you describe the floors in each of rooms?

Data Collection 2

Introduction: This research deals with online data communications. The participants will enter data into a Weblog server. The blog server is self contained on the data collection laptop machine as a local web server and is not posted on the Internet. The identities of the participants will not be revealed and the data entered into the Weblog server will not be used. Instead metadata about the Weblog server entries will be used in the research. Please be as deceptive as possible in your responses, use complete sentences, punctuation and use the spell checker/word replacement function where appropriate as you enter data.

Instructions: In the Weblog entry space provided a completely false description of your house. Below are some questions and prompts to help you think about some aspects of what to fabricate about. It is very important to be as deceptive as possible:

1. Can you describe the color scheme?
2. Describe the shingles or siding materials.
3. Describe the windows, are there many windows? For example are there picture windows installed or are they multi paned windows?
4. Please describe the landscaping, for example are there lots of trees, grass and shrubs?
5. Does your home have a backyard, please provide a description.
6. If there are trees what types of trees are on the property?
7. Please describe the interior of the house, what are the rooms like?
8. Please describe the style, for example Colonial, Split Level, Victorian, etc. Are there many rooms? Please describe the rooms.
9. Describe your furniture please. For example what is the upholstery like in your living room?
10. What types of window coverings do you have? Please describe them.
11. How would you describe the floors in each of rooms?

Appendix B

Performance Test Results

Table 4.

Test results "weblogentry" database table backup.

Test	Prototype Start Time	DB Backup Completion Time	Elapsed Time
1	May 25 16:19:45 EDT 2013	May 25 16:19:46 EDT 2013	1 Second
2	May 25 16:22:06 EDT 2013	May 25 16:22:07 EDT 2013	1 Second
3	May 25 16:24:06 EDT 2013	May 25 16:24:07 EDT 2013	1 Second
4	May 25 16:25:34 EDT 2013	May 25 16:25:35 EDT 2013	1 Second
5	May 25 16:32:16 EDT 2013	May 25 16:32:17 EDT 2013	1 Second
6	May 25 16:33:59 EDT 2013	May 25 16:33:60 EDT 2013	1 Second
7	May 25 16:35:07 EDT 2013	May 25 16:35:08 EDT 2013	1 Second
8	May 25 16:38:36 EDT 2013	May 25 16:38:37 EDT 2013	1 Second
9	May 25 16:40:01 EDT 2013	May 25 16:40:02 EDT 2013	1 Second
10	May 25 16:41:19 EDT 2013	May 25 16:41:20 EDT 2013	1 Second

Table 5.

Test results "vectors.csv" file creation.

Test	Prototype Start Time	"vectors.csv" File Creation	Elapsed Time
1	May 26 07:15:32 EDT 2013	May 26 07:15:47 EDT 2013	15 Seconds
2	May 26 07:19:52 EDT 2013	May 26 07:20:06 EDT 2013	14 Seconds
3	May 26 07:22:02 EDT 2013	May 26 07:22:16 EDT 2013	14 Seconds
4	May 26 07:25:42 EDT 2013	May 26 07:25:55 EDT 2013	13 Seconds
5	May 26 07:27:13 EDT 2013	May 26 07:27:27 EDT 2013	14 Seconds
6	May 26 07:28:49 EDT 2013	May 26 07:29:02 EDT 2013	13 Seconds
7	May 26 07:29:58 EDT 2013	May 26 07:30:12 EDT 2013	14 Seconds
8	May 26 07:31:07 EDT 2013	May 26 07:31:22 EDT 2013	15 Seconds
9	May 26 07:33:05 EDT 2013	May 26 07:33:19 EDT 2013	14 Seconds
10	May 26 07:34:06 EDT 2013	May 26 07:34:22 EDT 2013	16 Seconds

Table 6.

Test results backup "vectors.csv" file creation.

Test	**Backup "vectors.csv"**	**File Creation Backup "vectors.csv"**	**Elapsed Time**
1	May 26 07:19:32:05 EDT 2013	May 26 07:19:32:06 EDT 2013	1 Second
2	May 26 07:19:33:41 EDT 2013	May 26 07:19:33:42 EDT 2013	1 Second
3	May 26 07:19:34:59 EDT 2013	May 26 07:19:35:00 EDT 2013	1 Second
4	May 26 07:19:36:26 EDT 2013	May 26 07:19:36:27 EDT 2013	1 Second
5	May 26 07:19:37:40 EDT 2013	May 26 07:19:37:41 EDT 2013	1 Second
6	May 26 07:19:38:58 EDT 2013	May 26 07:19:38:59 EDT 2013	1 Second
7	May 26 07:19:40:04 EDT 2013	May 26 07:19:40:05 EDT 2013	1 Second
8	May 26 07:19:41:21 EDT 2013	May 26 07:19:41:22 EDT 2013	1 Second
9	May 26 07:19:42:35 EDT 2013	May 26 07:19:42:36 EDT 2013	1 Second
10	May 26 07:19:43:32 EDT 2013	May 26 07:19:43:33 EDT 2013	1 Second

Table 7.

Test results "vectors.csv" file deletion.

Test	**"vectors.csv" Deletion Time**	**Deletion of vectors.csv**	**Elapsed Time**
1	May 27 08:05:15 EDT 2013	May 27 08:05:16 EDT 2013	1 Second
2	May 27 08:06:29 EDT 2013	May 27 08:06:30 EDT 2013	1 Second
3	May 27 08:08:18 EDT 2013	May 27 08:08:19 EDT 2013	1 Second
4	May 27 08:09:23 EDT 2013	May 27 08:09:24 EDT 2013	1 Second
5	May 27 08:10:54 EDT 2013	May 27 08:10:55 EDT 2013	1 Second
6	May 27 08:12:48 EDT 2013	May 27 08:12:49 EDT 2013	1 Second
7	May 27 08:17:58 EDT 2013	May 27 08:17:59 EDT 2013	1 Second
8	May 27 08:19:10 EDT 2013	May 27 08:19:11 EDT 2013	1 Second
9	May 27 08:20:53 EDT 2013	May 27 08:20:54 EDT 2013	1 Second
10	May 27 08:22:16 EDT 2013	May 27 08:22:17 EDT 2013	1 Second

Appendix C

Neural Network File Division Procedures

Dividing files for testing

1. Launch the command line WEKA interface (CLI) and enter the following command without including the quotes:

"java weka.core.converters.CSVLoader filename.csv"

NOTE: filename.csv is the file that needs to be divided

2. After the command is run, copy and paste the output on in the CLI GUI into a notepad file and save as "all files" and put ".arff" extension on the filename.
3. Bring up WEKA explorer and open the .arff file that was just saved using the Open File button on the "Preprocess" tab.
4. Select randomize from the "Unsupervised" attribute dropdown list under the "Choose" button
5. Choose the "Apply" button
6. Select normalize from the "Unsupervised" attribute dropdown list under the "Choose" button
7. Choose the "Apply" Button
8. Select remove percentage from the "Unsupervised" attribute dropdown list under the "Choose" button.
9. Enter 20 percent
10. Choose the "Apply" button.
11. Save the file as the training.arf file by selecting the "Save" buton

12. Choose the “Undo” button Select invert selection “true” by clicking on the RemovePercentage text in the “Filter” field next to the “Choose” button. This brings up a GUI with an “inverSelection” drop down list. Default is “False”, change this to “True” by clicking the down arrow and click “OK”
13. Select “Apply”
14. Save the file as the testing.arff file.
15. Open the training.arf file on the preprocess tab using the “Open File” button.
16. Do not do any preprocessing
17. Open the testing.arf file on the classify tab using the “Use Test File” option.
18. Select start

Appendix D

Neural Network Validation Results

=== Run information ===

Scheme: weka.classifiers.functions.RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
Relation: 162_vec-weka.filters.unsupervised.attribute.Remove-R1-
weka.filters.unsupervised.instance.Randomize-S42-
weka.filters.unsupervised.instance.Normalize-N1.0-L2.0
Instances: 162
Attributes: 11
wc
comple
self
motion
sent
adj
uniq
compar
pos
analysis
TorF
Test mode: split 80% train, remainder test

Time taken to build model: 0.01 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances	27	81.8182 %
Incorrectly Classified Instances	6	18.1818 %
Kappa statistic	0.6374	
Mean absolute error	0.3023	
Root mean squared error	0.3746	
Relative absolute error	60.3401 %	
Root relative squared error	74.7378 %	
Total Number of Instances	33	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.778	0.133	0.875	0.778	0.824	F
0.867	0.222	0.765	0.867	0.813	T

=== Confusion Matrix ===

a	b	<-- classified as
14	4	a = F
2	13	b = T

Appendix E

Neural Network Test Plan, Test Procedures & Test Results

Neural Network Test Plan

Overview: Conduct 5 test runs using 212 samples on the dedicated test machine.

1. Randomize and Normalize the samples.
2. Split the samples into 2 files based on an 80-20 percent training testing split.
3. Load the training file into the WEKA Explorer Preprocess Tab.
4. Load the test file into the WEKA Explorer Classification Tab.
5. Select the kernel function (RBF).
6. Run the data.

Procedure:

1. Process the CSV file using the WEKA CLI to convert to an ARFF file type in accordance with the instructions.
2. Open the ARFF file using the WEKA Explorer “Open” button on the Preprocess tab.
3. Remove the Creator column from the predictor variable list by clicking the checkbox next to the Creator variable and clicking the “Remove” button.
4. Select randomize from the “Unsupervised” attribute dropdown list under the “Choose” button the select then “Apply”.

19. Select normalize from the “Unsupervised” attribute dropdown list under the “Choose” button.
20. Select “Apply”.

21. Select Remove Percentage from the “Unsupervised” attribute dropdown list under the “Choose” button.
22. Enter 20 percent in the text area.
23. Select Apply.
24. Save the file as training file using the Save button on the Preprocess tab.
25. Select the Undo button on the Preprocess tab.
26. Select invert selection by clicking on the text area and changing the “Invert Selection” value from False to True.
27. Select Apply.
28. Save file as the test file by selecting the Save button on the Preprocess tab.
29. Open the training file using the WEKA Explorer Open button on the Preprocess tab. Note: do not conduct further preprocessing.
30. On the classifier page select RBF under Functions.
31. Select Supplied Test Set.
32. Navigate to the test ARFF file in the file chooser box and select it.
33. Select the Start button on the Classify tab.
34. Analyze output.

Results: RBF runs

Run 1
=== Run information ===

Scheme: weka.classifiers.functions.RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1

Relation: vec_raw_20130610-weka.filters.unsupervised.attribute.Remove-R1-
weka.filters.unsupervised.instance.Randomize-S42-
weka.filters.unsupervised.instance.Normalize-N1.0-L2.0-
weka.filters.unsupervised.instance.RemovePercentage-P20
Instances: 170
Attributes: 11
WC_VEC
COMPLEXITY
SELF_REF_VEC
MOTION_VEC
SENT_COMP_VEC
ADJ_ADV_VEC
UNIQUE_VEC
COMPARE
POSTION
ANALYSIS
TorF
Test mode: user supplied test set: 42 instances

Time taken to build model: 0.02 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances	36	85.7143 %
Incorrectly Classified Instances	6	14.2857 %
Kappa statistic	0.7123	
Mean absolute error	0.3218	
Root mean squared error	0.3761	
Relative absolute error	64.3244 %	
Root relative squared error	75.1712 %	
Total Number of Instances	42	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.8	0.091	0.889	0.8	0.842	F
0.909	0.2	0.833	0.909	0.87	T

=== Confusion Matrix ===

a	b	<-- classified as
16	4	a = F
2	20	b = T

Run2
=== Run information ===

Scheme: weka.classifiers.functions.RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
Relation: vec_raw_20130610-weka.filters.unsupervised.attribute.Remove-R1-
weka.filters.unsupervised.instance.Randomize-S42-
weka.filters.unsupervised.instance.Normalize-N1.0-L2.0-
weka.filters.unsupervised.instance.RemovePercentage-P20
Instances: 170
Attributes: 11
WC_VEC
COMPLEXITY
SELF_REF_VEC
MOTION_VEC
SENT_COMP_VEC
ADJ_ADV_VEC
UNIQUE_VEC
COMPARE
POSTION
ANALYSIS
TorF
Test mode: user supplied test set: 42 instances

Time taken to build model: 0.01 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances	36	85.7143 %
Incorrectly Classified Instances	6	14.2857 %
Kappa statistic	0.7123	
Mean absolute error	0.3218	
Root mean squared error	0.3761	
Relative absolute error	64.3244 %	
Root relative squared error	75.1712 %	
Total Number of Instances	42	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.8	0.091	0.889	0.8	0.842	F
0.909	0.2	0.833	0.909	0.87	T

=== Confusion Matrix ===

```
 a  b   <-- classified as
16  4 |  a = F
 2 20 |  b = T
```

Run3

=== Run information ===

Scheme: weka.classifiers.functions.RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
Relation: vec_raw_20130610-weka.filters.unsupervised.attribute.Remove-R1-
weka.filters.unsupervised.instance.Randomize-S42-
weka.filters.unsupervised.instance.Normalize-N1.0-L2.0-
weka.filters.unsupervised.instance.RemovePercentage-P20
Instances: 170
Attributes: 11
WC_VEC
COMPLEXITY
SELF_REF_VEC
MOTION_VEC
SENT_COMP_VEC
ADJ_ADV_VEC
UNIQUE_VEC
COMPARE
POSTION
ANALYSIS
TorF
Test mode: user supplied test set: 42 instances

Time taken to build model: 0.01 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances	36	85.7143 %
Incorrectly Classified Instances	6	14.2857 %
Kappa statistic	0.7123	
Mean absolute error	0.3218	
Root mean squared error	0.3761	
Relative absolute error	64.3244 %	
Root relative squared error	75.1712 %	

Total Number of Instances 42

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.8	0.091	0.889	0.8	0.842	F
0.909	0.2	0.833	0.909	0.87	T

=== Confusion Matrix ===

a	b	<-- classified as
16	4	a = F
2	20	b = T

Run 4

=== Run information ===

Scheme: weka.classifiers.functions.RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
Relation: vec_raw_20130610-weka.filters.unsupervised.attribute.Remove-R1-weka.filters.unsupervised.instance.Randomize-S42-weka.filters.unsupervised.instance.Normalize-N1.0-L2.0-weka.filters.unsupervised.instance.RemovePercentage-P20
Instances: 170
Attributes: 11
WC_VEC
COMPLEXITY
SELF_REF_VEC
MOTION_VEC
SENT_COMP_VEC
ADJ_ADV_VEC
UNIQUE_VEC
COMPARE
POSTION
ANALYSIS
TorF
Test mode: user supplied test set: 42 instances

Time taken to build model: 0.01 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances	36	85.7143 %
Incorrectly Classified Instances	6	14.2857 %

Kappa statistic 0.7123
Mean absolute error 0.3218
Root mean squared error 0.3761
Relative absolute error 64.3244 %
Root relative squared error 75.1712 %
Total Number of Instances 42

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.8	0.091	0.889	0.8	0.842	F
0.909	0.2	0.833	0.909	0.87	T

=== Confusion Matrix ===

a	b	<-- classified as
16	4	a = F
2	20	b = T

Run 5

=== Run information ===

Scheme: weka.classifiers.functions.RBFNetwork -B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1
Relation: vec_raw_20130610-weka.filters.unsupervised.attribute.Remove-R1-
weka.filters.unsupervised.instance.Randomize-S42-
weka.filters.unsupervised.instance.Normalize-N1.0-L2.0-
weka.filters.unsupervised.instance.RemovePercentage-P20
Instances: 170
Attributes: 11
WC_VEC
COMPLEXITY
SELF_REF_VEC
MOTION_VEC
SENT_COMP_VEC
ADJ_ADV_VEC
UNIQUE_VEC
COMPARE
POSTION
ANALYSIS
TorF
Test mode: user supplied test set: 42 instances

Time taken to build model: 0.01 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances	36	85.7143 %
Incorrectly Classified Instances	6	14.2857 %
Kappa statistic	0.7123	
Mean absolute error	0.3218	
Root mean squared error	0.3761	
Relative absolute error	64.3244 %	
Root relative squared error	75.1712 %	
Total Number of Instances	42	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.8	0.091	0.889	0.8	0.842	F
0.909	0.2	0.833	0.909	0.87	T

=== Confusion Matrix ===

a	b	<-- classified as
16	4	a = F
2	20	b = T

Appendix F

ARFF NN Training File Contents

@relation vec_raw_20130610-weka.filters.unsupervised.attribute.Remove-R1-weka.filters.unsupervised.instance.Randomize-S42-weka.filters.unsupervised.instance.Normalize-N1.0-L2.0-weka.filters.unsupervised.instance.RemovePercentage-P20

@attribute WC_VEC numeric
@attribute COMPLEXITY numeric
@attribute SELF_REF_VEC numeric
@attribute MOTION_VEC numeric
@attribute SENT_COMP_VEC numeric
@attribute ADJ_ADV_VEC numeric
@attribute UNIQUE_VEC numeric
@attribute COMPARE numeric
@attribute POSTION numeric
@attribute ANALYSIS numeric
@attribute TorF {F,T}

@data

0.184977,0.97427,0.03053,0.00449,0.028734,0.00449,0.121223,0.00449,0.001796,0.008082,F
0.174373,0.978105,0.001155,0.00231,0.016167,0.003464,0.11086,0.005774,0.001155,0.017322,T
0.186009,0.974685,0.00248,0,0.017361,0.00496,0.119046,0,0,0.029761,T
0.186104,0.974891,0.01479,0.007395,0.013557,0.012325,0.118318,0.001232,0.001232,0.018487,F
0.190296,0.975266,0.010979,0.000915,0.022872,0.010064,0.108871,0.000915,0.00366,0.005489,F
0.186175,0.976571,0,0.00677,0.01354,0.010155,0.104935,0.00677,0.005077,0.015232,F
0.190689,0.973877,0.00681,0.011918,0.013621,0.010215,0.120883,0.001703,0.001703,0.010215,F
0.190799,0.9753,0.00678,0.002906,0.018402,0.008717,0.108474,0.003874,0,0.011622,F
0.173003,0.976959,0,0.009046,0.01583,0.009046,0.123251,0.003392,0.002261,0,F
0.183106,0.978054,0.014514,0.005582,0.01898,0.007815,0.094902,0.005582,0.005582,0.012281,T
0.178214,0.979021,0.010029,0.002314,0.018516,0.009258,0.094893,0.000771,0.001543,0.014658,F
0.157151,0.982373,0.001435,0.003588,0.014352,0.003588,0.099744,0,0.001435,0.007893,F
0.193542,0.972143,0.007387,0.013297,0.010342,0.011819,0.130013,0.002955,0.00591,0.007387,F
0.177498,0.977543,0.001305,0.005221,0.019577,0.007831,0.110936,0,0.003915,0.010441,F
0.175037,0.980724,0.004633,0.009267,0.014415,0.006693,0.0834,0.003089,0.006178,0.013385,F
0.180216,0.977747,0.002987,0.001991,0.009957,0.005974,0.105541,0.001991,0.00697,0.013939,T
0.169971,0.979315,0.003523,0.002642,0.029943,0.004403,0.104801,0.005284,0.005284,0.008807,F
0.177991,0.979438,0.004863,0.001945,0.021884,0.004863,0.089968,0.002432,0.003404,0.019453,T
0.172867,0.980711,0.012428,0.00339,0.015818,0.004519,0.086999,0.00226,0.007909,0.015818,T
0.181787,0.976313,0.027967,0.001271,0.017797,0.003814,0.110598,0.002542,0,0.02034,T
0.188057,0.979034,0.003039,0.0019,0.015196,0.002659,0.075223,0.00152,0.005319,0.013677,T
0.180126,0.975681,0.008339,0.001668,0.011675,0.010007,0.121752,0,0,0.021682,F
0.170555,0.97694,0.006822,0.010916,0.015009,0.016373,0.125529,0.002729,0.002729,0.008187,F
0.170292,0.979178,0.005322,0.002129,0.017029,0.006386,0.108561,0.002129,0.001064,0.00745,F
0.185381,0.976432,0.00283,0,0.022642,0.00283,0.106134,0.00566,0.007076,0.018397,T
0.173914,0.980243,0.001664,0.001664,0.009985,0.01165,0.092366,0.003328,0.004993,0.008321,T
0.169156,0.981798,0.000742,0.005688,0.018548,0.004204,0.083836,0.001731,0.002968,0.004946,F
0.180959,0.978007,0.001381,0.002763,0.019339,0.004144,0.10084,0.001381,0.005525,0.012432,F
0.176992,0.977477,0,0,0.020113,0.008045,0.11062,0.004023,0,0.022124,F
0.165662,0.981514,0.001661,0.002076,0.019099,0.005813,0.093418,0.003737,0.001246,0.006228,T
0.177067,0.979769,0.00154,0.003593,0.014884,0.007185,0.091356,0.001026,0.005132,0.006159,F
0.184009,0.976451,0.015719,0.002774,0.017569,0.005548,0.109111,0.001849,0.003699,0.012945,T

0.167873,0.982529,0.003634,0.000727,0.024709,0,0.074852,0.007267,0.00218,0.013081,F
0.189378,0.977367,0.002177,0.002177,0.015237,0.004354,0.090336,0.00653,0.00653,0.019591,T
0.178055,0.977409,0.005051,0.001263,0.029044,0.001263,0.107338,0.007577,0,0.02273,T
0.189981,0.976112,0.014558,0.001456,0.018197,0.003639,0.101906,0.002912,0.002912,0.012374,T
0.177485,0.980523,0.007527,0.000792,0.026543,0.003169,0.078838,0.003169,0.003566,0.007923,T
0.183726,0.97684,0.003638,0,0.009095,0,0.107325,0,0.009095,0.018191,F
0.161124,0.979577,0.005704,0.002852,0.019962,0.004278,0.118348,0.001426,0,0.001426,T
0.181365,0.978269,0.004122,0.005496,0.014427,0.007557,0.098239,0.004809,0.004122,0.009618,T
0.19288,0.973284,0.017765,0.001269,0.017765,0.008883,0.12055,0.003807,0.005076,0.015227,T
0.161668,0.980946,0.002431,0.004862,0.013371,0.013371,0.105753,0,0,0.006078,F
0.185351,0.975741,0.003972,0.003972,0.018535,0.00662,0.113859,0.003972,0,0.013239,F
0.169815,0.978853,0,0,0.042799,0.002761,0.103546,0.001381,0.004142,0.020709,T
0.14878,0.981537,0.002066,0.002066,0.024797,0.002066,0.115718,0.002066,0,0.020664,F
0.181668,0.977849,0.00158,0.00158,0.018957,0,0.101102,0.004739,0.006319,0.012638,F
0.186679,0.977776,0.006753,0.000965,0.017366,0.00193,0.092616,0.005789,0.004341,0.011095,T
0.183003,0.977597,0.004741,0.004741,0.018964,0.003793,0.101458,0.003793,0.003793,0.008534,F
0.160001,0.983088,0.001592,0.001592,0.019105,0.00199,0.086368,0.00199,0.00398,0.009154,T
0.164176,0.981815,0.006481,0.007561,0.015121,0.005401,0.092889,0.007561,0.00108,0.006481,F
0.179672,0.980077,0.001776,0.006344,0.019541,0.006598,0.081462,0.003045,0.004314,0.005837,F
0.158152,0.978047,0.004162,0,0.004162,0.004162,0.133181,0,0,0.024971,F
0.179081,0.974566,0.023358,0.003893,0.015572,0.003893,0.131067,0.003893,0.001298,0.011679,T
0.185819,0.979267,0.003252,0.001394,0.013472,0.001858,0.077115,0.004645,0.007897,0.016724,T
0.186534,0.978182,0,0.002985,0.008954,0.002238,0.085059,0.001492,0.002238,0.032084,T
0.164771,0.980386,0,0.002746,0.030208,0.002746,0.102982,0,0.002746,0.012358,T
0.175091,0.979842,0.003332,0.005182,0.014066,0.004812,0.094394,0.001851,0.003702,0.008144,F
0.193034,0.974267,0.017181,0.004043,0.018192,0.003032,0.113193,0.001011,0.005053,0.007075,T
0.168287,0.980452,0.003136,0,0.025086,0.007317,0.097209,0.006272,0.005226,0.013588,T
0.163914,0.979997,0.001744,0.001744,0.01395,0.005231,0.111601,0.005231,0,0.005231,F
0.185411,0.976087,0.013791,0.003065,0.021452,0.004597,0.108795,0.004597,0,0.018388,F
0.186291,0.977796,0.000927,0.003707,0.013902,0.004634,0.093609,0.001854,0.004634,0.013902,F
0.186711,0.97672,0.002008,0.007027,0.015057,0.011042,0.10239,0.003011,0.006023,0.015057,F
0.181616,0.977932,0.00508,0.00381,0.020321,0.00127,0.099063,0.00635,0.00635,0.017781,T
0.19245,0.973765,0.004935,0.001645,0.024673,0.008224,0.116786,0.00329,0.001645,0.019738,F
0.172498,0.98231,0.001291,0.00439,0.024015,0.003615,0.066882,0.003357,0.002841,0.014461,T
0.180776,0.978259,0.006198,0,0.02066,0.003099,0.098136,0.004132,0.005165,0.013429,T
0.177585,0.979134,0.011056,0.002073,0.016584,0.004837,0.096048,0.003455,0.002073,0.009674,F
0.174123,0.978451,0.011872,0.006925,0.020776,0.006925,0.107837,0.003957,0.000989,0.000989,T
0.183277,0.977073,0.002428,0.006069,0.021848,0.016993,0.104383,0.002428,0.003641,0.003641,F
0.145365,0.981214,0,0.004038,0.028265,0.002019,0.123156,0,0.006057,0.008076,T
0.185356,0.975396,0,0,0.018232,0.003039,0.115467,0.018232,0.003039,0.015193,F
0.176401,0.980709,0.003421,0.002443,0.023944,0.004886,0.078672,0.004886,0.001955,0.016125,T
0.178315,0.976101,0.013895,0.003474,0.019684,0.003474,0.121578,0.003474,0,0.005789,F
0.182256,0.977761,0.002293,0.001146,0.014901,0.001146,0.099725,0,0.003439,0.024071,T
0.176373,0.97887,0.00126,0.005039,0.016377,0.008819,0.100785,0.006299,0.00252,0.011338,T
0.19166,0.973045,0.008191,0.004914,0.009829,0.011467,0.126135,0.001638,0,0.014743,F
0.187133,0.976792,0.005141,0.002056,0.012338,0.003085,0.10282,0.001028,0.008226,0.005141,F
0.196323,0.974753,0,0.001373,0.028831,0.005492,0.098848,0.008237,0.001373,0.024712,F
0.166476,0.980196,0.00287,0,0.020092,0.008611,0.10333,0.005741,0.00287,0.017222,F
0.194504,0.97528,0.001379,0.009656,0.022071,0.013795,0.099321,0.005518,0.002759,0.017933,T
0.166816,0.982816,0.00175,0.000583,0.021581,0.002333,0.074659,0.005833,0.0035,0.012249,T
0.194148,0.977341,0.007924,0.001321,0.01893,0.006163,0.080565,0.003082,0.006163,0.010566,T
0.175217,0.975939,0.007536,0.011304,0.007536,0.011304,0.128115,0,0.001884,0.007536,F
0.174698,0.977536,0.011561,0.001285,0.016699,0.006423,0.115609,0.001285,0.001285,0.008992,T
0.191292,0.971884,0,0.003085,0.015427,0.009256,0.135755,0,0,0.009256,F
0.190461,0.97717,0.006962,0.004973,0.016908,0.006465,0.091501,0.002486,0.006465,0.006465,T
0.173935,0.981627,0.006246,0.004805,0.014414,0.003844,0.074955,0.001922,0.003363,0.015375,T

0.184443,0.975607,0.006472,0.006472,0.019415,0.011325,0.11649,0.003236,0,0.001618,F
0.182897,0.974616,0,0,0.012527,0.005011,0.127777,0.007516,0.002505,0.010022,T
0.174876,0.976209,0.011068,0.004427,0.013282,0.008854,0.126177,0.004427,0.004427,0.008854,T
0.180731,0.978217,0,0.006484,0.021882,0.006484,0.098875,0,0.004863,0.008105,F
0.179962,0.977213,0.020318,0.001935,0.020318,0.004838,0.108364,0.00387,0.005805,0.004838,T
0.1747,0.980382,0.001474,0.003686,0.013268,0.008846,0.089193,0.003686,0.003686,0.008846,T
0.176134,0.981744,0.004609,0.004609,0.013827,0.013827,0.068149,0.003951,0.00428,0.007243,F
0.183513,0.978431,0,0.001844,0.014755,0.003689,0.089451,0.007377,0.009222,0.024899,T
0.180062,0.980455,0.006064,0.007382,0.013709,0.005273,0.076718,0.003427,0.003164,0.008436,T
0.198347,0.970177,0.025871,0.005749,0.01581,0.012936,0.135106,0,0.002875,0.005749,F
0.197326,0.975778,0.000987,0.004933,0.015786,0.00592,0.089783,0.009866,0.007893,0.019733,F
0.182684,0.976414,0.011024,0,0.012599,0.004725,0.11339,0.006299,0.001575,0.006299,F
0.168944,0.977864,0,0.002238,0.015664,0.002238,0.121953,0.004475,0.005594,0.007832,F
0.178426,0.977528,0.01525,0.001525,0.016775,0.004575,0.108275,0.001525,0.007625,0.016775,T
0.182747,0.976929,0.005124,0.003416,0.020495,0.003416,0.107599,0.003416,0.003416,0.011955,F
0.191111,0.974666,0.001365,0.004095,0.017746,0.004095,0.113302,0.00546,0.004095,0.016381,T
0.179111,0.975973,0.005483,0,0.010966,0.005483,0.122454,0.001828,0.007311,0.012794,T
0.17761,0.975704,0,0.004613,0.02768,0.004613,0.124558,0,0,0.011533,T
0.183645,0.977606,0,0.0011,0.024193,0.004399,0.096771,0.0011,0.002199,0.024193,T
0.190782,0.975667,0.004184,0.009204,0.015062,0.004184,0.105432,0.003347,0.006694,0.012551,F
0.174698,0.975958,0.010079,0.00336,0.010079,0.005039,0.129344,0.00168,0.00168,0.00336,F
0.183249,0.974826,0.013518,0.001502,0.022531,0.01502,0.123168,0.003004,0,0.006008,F
0.189828,0.977722,0.008629,0.000539,0.016718,0.004854,0.086825,0.002696,0.005932,0.008089,T
0.187302,0.978507,0.00084,0.00084,0.020998,0.00504,0.081472,0.00084,0.00504,0.017638,T
0.177841,0.982213,0.000303,0.002424,0.027267,0.001818,0.050292,0.009695,0.005756,0.014845,T
0.178114,0.982053,0.001212,0.003938,0.018478,0.003635,0.056342,0.004544,0.005755,0.015752,F
0.174226,0.976323,0,0.004931,0.01808,0.004931,0.124917,0.001644,0,0.021367,T
0.191027,0.972979,0.014694,0.006298,0.017843,0.004198,0.127002,0.00105,0.003149,0.009446,F
0.132999,0.984195,0,0,0.035467,0.004433,0.110833,0.001478,0,0.010344,T
0.167534,0.978637,0.005166,0.005166,0.020665,0.006642,0.11661,0.00369,0.000738,0.008118,T
0.180711,0.97711,0.005449,0.001816,0.033599,0.007265,0.106247,0.002724,0.002724,0.009081,T
0.169546,0.977485,0.01384,0.00173,0.024221,0.01038,0.121104,0.00865,0,0.01211,F
0.17918,0.97774,0.01736,0.00372,0.02232,0.0031,0.10478,0.00496,0.00248,0.0093,T
0.189502,0.973702,0.004622,0.004622,0.010785,0.010785,0.124794,0.009244,0,0.007703,F
0.177557,0.977298,0.002935,0.002935,0.011739,0.00587,0.114458,0.002935,0.002935,0.007337,F
0.16899,0.978093,0.00256,0.00256,0.023044,0.00256,0.117781,0.00256,0.005121,0.017923,T
0.16704,0.975962,0.001877,0.007507,0.011261,0.003754,0.138887,0.001877,0.003754,0.009384,F
0.179509,0.975333,0.003989,0.003989,0.013962,0.005984,0.125656,0.003989,0.005984,0.019945,T
0.175292,0.978242,0.005655,0.002262,0.016964,0.003393,0.108568,0.005655,0.002262,0.01244,T
0.193153,0.977873,0.004613,0.013261,0.012685,0.014414,0.076108,0.000577,0.005766,0.008072,F
0.178133,0.977366,0.001576,0.006306,0.011035,0.011035,0.111924,0,0.003153,0.014188,F
0.185807,0.978352,0.012616,0.003441,0.012043,0.006308,0.087742,0,0.005161,0.01491,T
0.174238,0.977763,0.001692,0.006767,0.008458,0.01015,0.115031,0.005075,0,0.011841,F
0.187801,0.977112,0,0.002722,0.018145,0.002722,0.096169,0.004536,0.006351,0.018145,T
0.178398,0.975495,0.010122,0.006326,0.018979,0.006326,0.126523,0,0.00253,0.005061,F
0.191629,0.976462,0.002818,0.001409,0.019727,0,0.092996,0.004227,0.004227,0.026772,T
0.184127,0.976299,0.018555,0,0.012846,0.007137,0.109905,0.005709,0.004282,0.015701,T
0.185768,0.975283,0.015481,0,0.012041,0.00516,0.116965,0.0086,0.00172,0.012041,T
0.182387,0.976306,0.004598,0.004598,0.013794,0.001533,0.113417,0.001533,0,0.021457,T
0.186053,0.977841,0.008496,0.003398,0.016991,0.001699,0.092602,0.005097,0.005097,0.014442,T
0.184968,0.975592,0.006767,0.009023,0.014662,0.006767,0.116169,0.006767,0.005639,0.006767,F
0.156872,0.981774,0.00235,0.006463,0.019389,0.011163,0.104582,0.001763,0,0.0047,F
0.170382,0.979412,0.005642,0.004513,0.019182,0.004513,0.106065,0,0.003385,0.004513,F
0.136746,0.977733,0,0,0.068373,0.006837,0.143583,0,0,0,F
0.178449,0.980054,0.002266,0.0017,0.015862,0.002266,0.084976,0.002833,0.003966,0.011897,T
0.186178,0.979159,0.001217,0.004867,0.01298,0.003651,0.077067,0.003245,0.009735,0.018253,T

0.167717,0.978135,0.00128,0.005121,0.014083,0.008962,0.121627,0.003841,0.00128,0.002561,F
0.182579,0.966597,0.01074,0,0.01074,0.01611,0.177209,0,0,0.02148,F
0.180581,0.978659,0.00153,0.000765,0.019129,0.011478,0.094882,0.003061,0.002296,0.009947,F
0.150579,0.97974,0.001956,0.001956,0.044978,0.005867,0.123201,0,0.001956,0.013689,F
0.189529,0.975157,0.004585,0,0.01987,0.001528,0.110049,0.009171,0.003057,0.022927,T
0.167575,0.978321,0.004788,0.003192,0.030323,0.004788,0.116505,0.006384,0.003192,0.014364,F
0.186336,0.977002,0.008058,0.007051,0.01813,0.004029,0.100722,0.003022,0.003022,0.011079,T
0.190903,0.978179,0.011631,0.004813,0.017245,0.006016,0.077404,0.002406,0.008823,0.012433,T
0.17333,0.980179,0.003041,0.006082,0.013177,0.017232,0.09224,0.004055,0.004055,0.012164,F
0.177279,0.975947,0.009138,0.001828,0.016449,0.005483,0.124278,0.00731,0,0.014621,F
0.184698,0.975494,0.007173,0.006276,0.024208,0.006276,0.115661,0.00269,0.010759,0.008966,F
0.170201,0.977695,0.016636,0,0.017916,0.003839,0.119013,0,0,0.019196,T
0.18699,0.977152,0.005821,0.00291,0.0211,0.005821,0.096769,0.002183,0.003638,0.017462,T
0.186009,0.977363,0,0.004895,0.014685,0.003263,0.097899,0.001632,0.001632,0.017948,F
0.147614,0.981804,0,0.003433,0.020597,0.006866,0.113285,0,0,0.030896,F
0.168134,0.980784,0,0.004873,0.026804,0.017057,0.092596,0.012184,0.001218,0.006092,F
0.175919,0.975553,0.001999,0,0.015993,0.007996,0.129941,0.001999,0.005997,0.009995,T
0.180907,0.978123,0.002044,0.006132,0.013287,0.00511,0.100163,0.00511,0.003066,0.015331,T
0.194302,0.974065,0.008948,0.005113,0.01534,0.010226,0.113769,0.005113,0.003835,0.003835,F
0.183423,0.979426,0.006533,0.006282,0.018342,0.008292,0.080907,0.002261,0.001508,0.006282,T
0.169172,0.982727,0.008325,0.001332,0.019648,0.007326,0.070599,0.00333,0.002997,0.010656,T
0.184698,0.975714,0.028023,0.001274,0.019107,0.003821,0.110819,0.002548,0,0.02038,T
0.16855,0.977588,0.002931,0.007328,0.014656,0.007328,0.12458,0.002931,0,0.007328,F
0.183507,0.975166,0.0091,0,0.012133,0.006066,0.122844,0.003033,0,0.003033,F
0.178782,0.977117,0.011244,0.001124,0.013493,0.007871,0.112442,0.003373,0.002249,0.015742,T
0.1814,0.978204,0.000848,0.001695,0.016953,0.007629,0.098329,0.005934,0.004238,0.011867,F

Appendix G

ARFF NN Testing File Contents

@relation vec_raw_20130610-weka.filters.unsupervised.attribute.Remove-R1-weka.filters.unsupervised.instance.Randomize-S42-weka.filters.unsupervised.instance.Normalize-N1.0-L2.0-weka.filters.unsupervised.instance.RemovePercentage-P20-V

@attribute WC_VEC numeric
@attribute COMPLEXITY numeric
@attribute SELF_REF_VEC numeric
@attribute MOTION_VEC numeric
@attribute SENT_COMP_VEC numeric
@attribute ADJ_ADV_VEC numeric
@attribute UNIQUE_VEC numeric
@attribute COMPARE numeric
@attribute POSTION numeric
@attribute ANALYSIS numeric
@attribute TorF {F,T}

@data

0.191192,0.973339,0.003862,0.003862,0.011587,0.007725,0.123599,0.001931,0.003862,0.023175,F
0.177164,0.978292,0.001728,0.004321,0.012963,0.00605,0.106298,0.005185,0.001728,0.001728,F
0.190216,0.974611,0.013727,0.001961,0.015688,0.003922,0.115698,0.007844,0.003922,0.005883,F
0.176227,0.978906,0.016094,0.004426,0.016899,0.010059,0.099379,0.002816,0.000805,0.011668,F
0.176042,0.982319,0.001326,0.003315,0.023207,0.003978,0.057354,0.007294,0.006631,0.010277,T
0.190816,0.97463,0.023485,0.001468,0.019082,0.001468,0.111554,0.001468,0.002936,0.017614,T
0.177191,0.980216,0,0.004176,0.023267,0.00358,0.084121,0.00179,0.007756,0.008352,T
0.1869,0.976036,0.028754,0.007987,0.01278,0.014377,0.105431,0.001597,0.001597,0.00639,F
0.189189,0.976957,0.013957,0.004652,0.015507,0.003877,0.094594,0.003877,0.012406,0.013181,T
0.171452,0.979728,0.005878,0.008818,0.010777,0.007838,0.101892,0.00098,0.001959,0.007838,F
0.191179,0.973275,0.01264,0,0.04266,0.00632,0.118499,0.00158,0,0.01106,F
0.169765,0.978549,0.022422,0.009609,0.014414,0.009609,0.112109,0.003203,0.004805,0.011211,T
0.181794,0.978008,0.004602,0.002301,0.019944,0.012273,0.098951,0.002301,0.004602,0.007671,T
0.165067,0.980088,0.003439,0.001146,0.013756,0.001146,0.108899,0,0.003439,0.010317,T
0.190819,0.974837,0.004148,0.002074,0.010371,0.004148,0.112003,0,0.008296,0.022815,T
0.16541,0.981812,0.00071,0.008519,0.012778,0.005679,0.091579,0.00071,0.00213,0.004969,F
0.184193,0.976905,0.004093,0.001364,0.016373,0.004093,0.105058,0.008186,0.002729,0.017737,T
0.18286,0.981347,0,0.001925,0.024381,0.003529,0.051329,0.006095,0.008983,0.012511,F
0.184572,0.976178,0.011279,0.002051,0.017432,0.001025,0.111768,0.002051,0.003076,0.008203,T
0.178169,0.980399,0.00468,0.006865,0.016226,0.005617,0.080504,0.003432,0.004368,0.014041,T
0.183325,0.978262,0.00158,0.002371,0.018965,0.006322,0.094033,0,0.002371,0.011853,T
0.181206,0.979464,0.00317,0.003698,0.015321,0.003698,0.085584,0.002113,0.011094,0.009509,T
0.184396,0.975667,0.002334,0,0.023341,0.002334,0.114372,0.009337,0,0.018673,F
0.178878,0.980118,0.003038,0.005738,0.020588,0.008775,0.082352,0.003713,0.0027,0.005063,T
0.188902,0.977848,0.004167,0.002778,0.01389,0.004167,0.087506,0.009723,0.009723,0.006945,T
0.193285,0.975096,0.008673,0.004956,0.016107,0.003717,0.106554,0.006195,0.004956,0.006195,F
0.174628,0.976398,0.006074,0.006074,0.012148,0.004556,0.126036,0.001519,0.001519,0.004556,F
0.172586,0.976474,0.004542,0.004542,0.021195,0.003028,0.127169,0.001514,0.001514,0.006056,F
0.176158,0.976528,0.005744,0.001915,0.015318,0.001915,0.12063,0.001915,0.001915,0.022977,T
0.187711,0.976921,0.00137,0.006851,0.019182,0.013702,0.097281,0.009591,0.00137,0.015072,F
0.174574,0.978743,0.002253,0.002253,0.015768,0.00901,0.105871,0.002253,0.001126,0.005631,F
0.183255,0.978085,0,0.00759,0.018434,0.011928,0.095423,0.002169,0.003253,0.010301,F

0.181057,0.976755,0.007147,0,0.019059,0.002382,0.11197,0.002382,0,0.014294,T
0.174812,0.97808,0.007224,0.002889,0.008668,0.011558,0.111244,0.010113,0.001445,0.007224,F
0.192695,0.97375,0.012846,0.001285,0.020554,0.007708,0.115617,0.005139,0.006423,0.024408,T
0.168334,0.977938,0,0.004008,0.014028,0,0.120238,0.004008,0.006012,0.024048,T
0.181618,0.979463,0,0.007067,0.01484,0.011307,0.084802,0.00212,0.003533,0.007774,F
0.173387,0.978309,0,0.003797,0.016453,0.003797,0.110107,0.003797,0.001266,0.02025,T
0.180531,0.979683,0.004814,0.005617,0.021664,0.006419,0.081841,0.004814,0.001605,0.018454,T
0.168808,0.979505,0.006252,0,0.012504,0.004168,0.106287,0.006252,0,0.022925,F
0.181098,0.977504,0,0.002118,0.021181,0.002118,0.102728,0.005295,0.002118,0.025417,T
0.176441,0.975275,0.001939,0.003878,0.019389,0.01745,0.129907,0.005817,0.001939,0.009695,F

References

Agarwal, B., Gupta, M., & Tayal, S. (2009). *Software engineering and testing: An introduction*. Sudbury, MA: Jones and Bartlett Publishers.

Agent Oriented Software (2011). *JACK intelligent agents agent manual*. JACK Autonomous Software.

Apache Roller (2011). *Apache roller Weblogger version 5.0 installation guide*. The Apache Software Foundation.

Apache Tomcat 7 (2011). *Tomcat user guide*. The Apache Software Foundation.

Azab, M., & Eltoweissy, M. (2011). Towards a cooperative autonomous resilient defense platform for cyber-physical systems. *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research,* New York, NY, 1-4.

Balke, T. De Vos, M., Padget, J., Traskas, D.(2011). On-line reasoning for institutionally-situated BDI Agents. *The 10th International Conference on Autonomous Agents and Multiagent Systems*, 3, 1109-1110.

Boongoen, T. & Shen, Q. (2009, June). Intelligent hybrid approach to false identity detection. *Proceedings of the 12th ACM International Conference on Artificial Intelligence and Law*, Barcelona, Spain, 147-156.

Elagouni, K., Garcia, C. & Sebillot, P. (2011). A comprehensive neural-based approach for text recognition in videos using natural language processing. *Proceedings of the 1st ACM International Conference on Multimedia Retrieval,* Trento, Italy, (1-8).

Fette, I., Sadeh, N., & Tomasic, A. (2007). Learning to detect phishing emails. *Proceedings of the 16th ACM International Conference on World Wide Web*, Banff, Alberta, Canada, 649-656.

Filippidis, A., Russo, M., & Jain, L. (2010). Novel extensions of ART2 in surface landmine detection. In Jain, L., Lazzerini, B. & Halici (Eds.), *Studies in fuzziness and Soft Computing: Innovations in ART Neural Networks* (pp. 1-15). New York, NY: Springer Verlag.

Frank, L. (2008). A transaction model for mobile transactions. *22nd IEEE International Conference on Advanced Information Networking and Applications - Workshops*, Ginowan, Okinawa, Japan, 868-873.

Gandotra, V., Singhal, A., & Bedi, P. (2011). Layered security architecture for threat management using multi-agent system. *ACM SIGSOFT Software Engineering Notes,* 36(5), 1-11.

Hall, M., Eibe F., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. (2009). *The WEKA data mining software: An update*. ACM SIGKDD Explorations, 11(1), 10-18.

Heaton, J. (2011). *Programming neural networks with ENCOG3 in Java (3rd ed.)* St. Louis, MO: Heaton Research, Inc.

Hsiao, C. (2006). ARTMAP [Software]. Available from http://cns.bu.edu/~gsc/CN710/pmwiki.php?n=Main.ARTMAPWithWekaInterface.

Jensen, M., Burgoon J., & Nunamaker, Jr., J. (2010). Judging the credibility of information gathered from face to face interactions. *ACM Journal of Data and Information Quality*, 2(1), 1-20.

Kolo, B., (2010). *Binary and multiclass classification.* Lexington, KY: Weatherford Press.

Kuderna, B., Hoszu, A., Vacariu, L., & Cret, O. (2009). Agent based smart house platform with affective control. *Proceedings of the1st ACM Workshop on Information Security Governance,* Prague, CZ, 1-7.

Lau, R., Liao, S., Kwok, R., Xu, K., Xia, Y, & Li, Y. (2012). Text mining and probabilistic language modeling for online review of spam detection. *ACM Transactions on Management Information Systems*, 2(4), 1-30.

Lim, Y., Cheng, P., Rohatgi, P., & Clark, J. (2009). Dynamic security policy learning. *Proceedings of the1st ACM Workshop on Information Security Governance,* New York, NY, 39-48.

Ma, B., Chen, B., Bai, X., & Huang, J. (2010). Design of BDI agent for adaptive performance testing of web services. *10th IEEE Conference on Quality Software*, Zhangjiajie, Hunan, China, 435-440.

Machorro-Fernandez, F., Para-Vega, V., & Lopez-Juarez, I. (2010). Human training using HRI approach based on fuzzy ARTMAP networks. *Proceedings of the5th ACM/IEEE International Conference on Human-robot Interactions.* Saltillo, Mexico, 113-114.

Maranzato, R., Periera, A., Neubert, M., & Periera do Lago, A. (2010). Fraud detection in reputation systems in e-markets using logistic regression and stepwise optimization. *ACM SIGAPP Applied Computing Review,* 11(1), 14-26.

Meneguzzi, F., & Luck, M. (2009). Norm-based behavior modification in BDI agents. *Proceedings of the 8th ACM International Conference on Autonomous Agents and Multiagent Systems*, Budapest, Hungary, 177-184.

Miles, J,. & Tashakkori, R. (2010). Improving believability of simulated characters. *Journal of Computing Sciences in Colleges, 25(3), 32-39.*

Murach, J. (2012). *Murach's MySQL.* Fresno, CA: Mike Murach & Associates, Inc.

Nguyn, C., Perinie, A., Tonella, P., Miles, S., Harmon, M., & Luck, M. (2009, May). Evolutionary testing of autonomous software agents. *Proceedings of the 8th ACM International Conference on Autonomous Agents and Multiagent Systems*, Budapest, Hungary, 521-528.

Oong, T., & Isa, N., (2012). Multilayer Fuzzy ARTMAP; Fast learning and fast testing for pattern classification. *Proceedings of the 27th Annual ACM symposium on Applied Computing*, Riva del Garda, Italy, 27-32.

Patel, P., & Hexmoor, H. (2009). Designing BOTs with BDI agents. *2009 IEEE International Symposium on Collaborative Technologies and Systems*, Baltimore, Maryland, USA, 180-186.

Phua, C., Lee, V., Smith-Miles, K., & Gayler, R. (2007). Adaptive communal detection in search of adversarial identity crime. *Proceedings of the 2007 ACM International Workshop on Domain Driven Data Mining,* San Jose, California, USA, 1-10.

Ross, T. (2011). *Fuzzy logic with engineering applications (3rd ed.)* West Sussex, UK: John Wiley & Sons, LTD.

Rubin, V. (2008). On deception and deception detection: Content analysis of computer-mediated stated beliefs. *Proceedings of the 73rd ACM American Society for Information Science & Technology,* Silver Springs, MD, USA, 22-27.

Santos, E. & Deging, L. (2010). On deception detection in multi-agent systems. *IEEE Transactions on Systems, Man and Cybernetices,* 40(2), 224-235.

Schildt, H. (2011). *Java the complete reference (8th ed.)* New York, NY: McGraw-Hill Companies.

Sherrod, P. (2013). *DTREG Predictive Modeling Software User Guide*. DTREG Predictive Modeling Software.

Sloan, C., Kelleher, J., & Mac Namee, B. (2011). Feasibility study of utility-directed behavior for computer game agents. *Proceedings of the 8th International*

Conference on Advances in Computer Entertainment Technology, New York, NY, 1-6.

Taghezout, N., & Adla, A. (2008, December). Proposal for an adaptive user interface design: A hybrid approach: IDSS and BDI agents, application to the boiler combustion management system (GLZ). *2008 IEEE Advanced Software Engineering and its Applications*, Hainan Island, China, 91-94.

Tan, A. (2010). Supervised adaptive resonance theory and rules. In Jain, L., LAzzerinin, B. & Halici (Eds.), *Studies in fuzziness and Soft Computing: Innovations in ART Neural Networks* (pp. 55-86). New York, NY: Springer Verlag.

Thangarajah, J., Harland, J., Morley, D., & York-Smith, N. (2008, May). Suspending and resuming tasks in BDI agents. *Proceedings of the 7th ACM International Conference on Autonomous Agents and Multiagents*, Honolulu, Hawaii, USA, 405-412.

Telang, P., Meneguzzi, F. & Singh M. (2013). Hierarchical planning about goals and commitments. *Proceedings of the 2013 International Conference on Autonomous agents and Multi-agent Systems*, Richland, SC, USA, 877-884.

Thimm, M., & Krumpelmann, P. (2009). Know-how for motivated BDI agents. *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, Budapest, Hungary, 1143-1144.

Toma, C., & Hancock, J., (2010). Reading between the lines: Linguistic cues to deception in online dating profiles. *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, Savannah, Georgia, USA, 5-8.

Warkentin, D., Woodworth, M., Hancock, J., & Cormier, N. (2010). Warrants and deception in computer mediated communication. *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, Savannah, Georgia, USA, 9-12.

Whitten, I., Eibe, F., & Hall, M. (2011). *Data mining (3rd ed.)* Burlington, MA: Morgan Kaufman Publishers.

U.S. Department of Defense, Joint Chiefs of Staff (2006). *Information Operations* (Joint Publication 3-13).

U.S. Department of Defense, (2011). Strategy for Operating in Cyber Space.

Zhou, L., & Zhang D. (2008, September). Following linguistic footprints *Communications of the ACM*, 51(9), 119-122.