Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Allenoush Hayrapetian

Entitled

ANALYZING AND EVALUATING SECURITY FEATURES IN SOFTWARE REQUIREMENTS

For the degree of Master of Science

Is approved by the final examining committee:

Dr. Rajeev R. Raje
Chair

Dr. Mihran Tuceryan

Dr. Yao Liang

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Dr. Rajeev R. Raje

Approved by: Dr. Shiaofen Fang 11/1/2016

Head of the Departmental Graduate Program Date

ANALYZING AND EVALUATING SECURITY FEATURES IN SOFTWARE REQUIREMENTS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Allenoush Hayrapetian

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2016

Purdue University

Indianapolis, Indiana

I am dedicating my thesis to my kind father to whom I owe my self-confidence and to my dear mother who is a true example of learning continuously.

ACKNOWLEDGEMENTS

Above everything I am so thankful to God who gifted me with life and all of the beauty of it. If I have the ability of doing anything, it is because of His grace and love.

I am so thankful to my friends and family who always believe in me and my abilities and have always been supportive of me to accomplish my goals.

I am very grateful to my professor and advisor Dr. Rajeev Raje, who always trusted my abilities and encourages me to push myself to my full potential.

I would like to thank Dr. Mihran Tuceryan and Dr. Yao Liang for accepting to be members of my thesis committee.

I would like to thank the Computer Science department for providing the opportunity for high quality study and research.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

ABSTRACT

Hayrapetian, Allenoush. M.S., Purdue University, December 2016. Analyzing and Evaluating Security Features in Software Requirements. Major Professor: Dr. Rajeev Raje.

Software requirements, for complex projects, often contain specifications of non-functional attributes (e.g., security-related features). The process of analyzing such requirements for standards compliance is laborious and error prone. Due to the inherent free-flowing nature of software requirements, it is tempting to apply Natural Language Processing (NLP) and Machine Learning (ML) based techniques for analyzing these documents. In this thesis, we propose a novel semi-automatic methodology that assesses the security requirements of the software system with respect to completeness and ambiguity, creating a bridge between the requirements documents and being in compliance.

Security standards, e.g., those introduced by the ISO and OWASP, are compared against annotated software project documents for textual entailment relationships (NLP), and the results are used to train a neural network model (ML) for classifying security-based requirements. Hence, this approach aims to identify the appropriate structures that underlie software requirements documents. Once such structures are formalized and empirically validated, they will provide guidelines to software organizations for

generating comprehensive and unambiguous requirements specification documents as related to security-oriented features. The proposed solution will assist organizations during the early phases of developing secure software and reduce overall development effort and costs.

CHAPTER 1. INTRODUCTION

1. Motivation

A successful realization of large, complex, and software-intensive systems requires the usage of best practices during the entire software life-cycle, including the initial phase of requirements analysis.

Among non-functional requirements of software which identify the quality of software, security is one of the most important features. Complete and unambiguous security requirements will result in high quality software with minimum security vulnerabilities. Identifying the level of security features to be considered in the requirements analysis phase of software has four major benefits.

First, identifying the completeness level of security requirements before deploying the project results in the reduction of defects. Determining the adequacy of security requirements which an organization can perform manually or through some automated process, provides a fundamental basis for estimating costs, defining the scope of the project, constructing the design and testing specifications, and helping understand the possible consequences of either a successful or unsuccessful deployment. As a result, the quality of the requirements gathering phase is related to the success of the software project. If the requirements are ambiguous, it will result in more software defects.

In order to ensure that the requirements are complete, the process for gathering both business and technical requirements needs to be approached in that respective order and carried out methodically (Javed, Maqsood, & Durrani, 2004).

Secondly, it results in the early discovery of errors. The earlier the software vulnerabilities (e.g., incompleteness) are discovered in the life cycle of a project, the earlier they can be corrected and thus, the costs are decreased in future phases.

The below graph (Ambler, n.d.; Karg & Beckhaus, 2008) clearly demonstrates the cost associated with creating poor quality software, and the sooner it is discovered, the less costly it will be to change it.

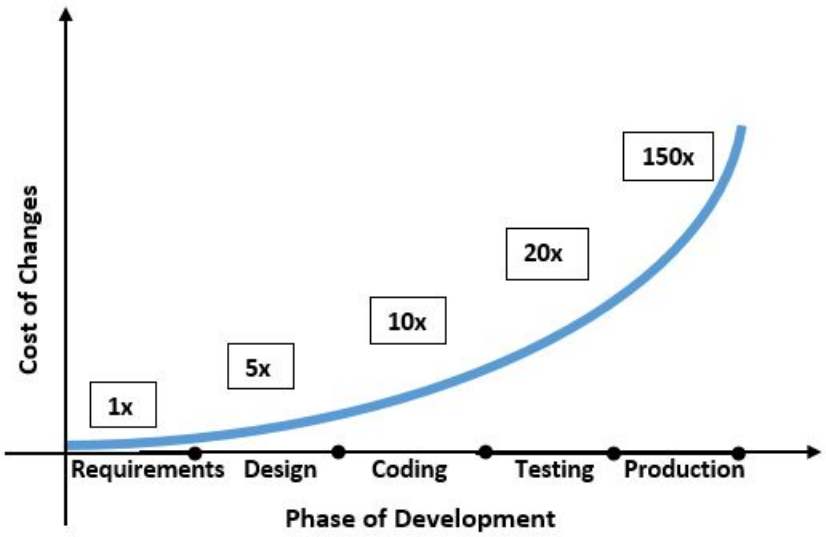

Figure 1.1 Cost of Quality

Thirdly, assessing software requirements against specifically defined security standards, such as security standards introduced by the International Organization for Standardization (ISO) and the Open Web Application Security Project (OWASP), helps in

generating guidelines for the future creation of requirements, and therefore, results in standardization.

Finally, this work will not only be beneficial for the software providers, but also for those who have a vested interest in the development and the outcome of project (i.e. stakeholders) who may not have enough technical knowledge about the security risks involved in software, therefore enabling them to identify and request the security features that they are looking for. Stakeholders may have a general idea about the risks involved in software but not be familiar enough with the specifics of security features that a developer could take and implement. For example, they would not want unauthorized access to their software, but they may not know about the brute force tactics which may enable adversaries to access their system and the ways to prevent such access. Therefore, standards such as OWASP and ISO should be considered in order to define all of the features that may be involved in security-related aspects of a software system (Malhotra, Chug, Hayrapetian, & Raje, 2016).

These benefits do emphasize the need for a formal technique to analyze and evaluate security requirements. Rojas and Sliesarieva (2010) indicate that the desirable qualities of software requirements are accuracy, verifiability, and unambiguity.

Accordingly, this thesis aims to identify the degree of incompleteness and ambiguity of security requirements, creating a bridge between the requirements and being in compliance with the security standards before moving to the next phase of software development.

1.1 Aim of Work

The goal of this thesis is to analyze the set of security requirements for any given software project and to provide feedback about its completeness and inherent ambiguity when evaluated with respect to a given security standard. Complex, and often distributed, projects present a myriad of challenges to stakeholders, such as the proper discovery and utilization of domain experts, designers, developers, testers, and users. The process of analyzing these requirements is laborious and requires a large amount of manual intervention and hence, is error prone. Due to the inherent free-flowing nature of software requirements, it is tempting to apply NLP-based techniques for achieving the necessary analysis.

In this thesis, a semi-automatic method is devised that can assess the completeness and ambiguity of software documents with respect to certain security features. Security standards introduced by the ISO and OWASP along with manually annotated project requirements documents, are used to construct a model using NLP-based and machine learning (ML) techniques. This method is used to analyze the level of completeness of the given security requirements document and thus, to identify its vulnerability with respect to certain security features at an earlier phase of software development.

The proposed approach identifies the appropriate structures that underlie software requirements documents. Once such structures are formalized and empirically validated, they will help various organizations to create guidelines for generating comprehensive and unambiguous requirements documents. Regardless of the software methodology (e.g., Agile or Waterfall) being used by organizations, this method for analyzing security

features in requirements documents can be beneficial to them as it will allow a continuous analysis and enhance security requirements in their software projects. Hence, the results of this thesis can assist organizations during the early phases of developing secure software, and thereby, reduce overall development costs, and result in secure software projects (Raje & Malhotra, 2015).

In summary, the goals of this thesis are as follows:

- To compile a gold standard for software security requirements documents
- To analyze software security requirements documents against a gold standard for semantic relationships
- To provide feedback about the completeness and ambiguity of a software security requirements document with respect to the gold standard

This thesis makes three main contributions: i) a generalized architecture for semantic analysis, ii) a compiled software security gold standard, iii) an algorithm for interpreting semantic classification with respect to the completeness of a given security requirements document.

First, the generalized architecture devised in this thesis provides the benefit of extendibility both within and outside of the given domain. The current features of this thesis include assessing the completeness and ambiguity of a given software requirements document with respect to security, however, it could also be expanded to include contradictions or inconsistency, for example. In addition to assessing security features, other features can be added as well, such as usability and maintainability. The

core components of the system can be reused in any other domain to accomplish similar goals in determining the semantic relationships between statements.

Secondly, a gold standard was developed through gathering data from ISO, OWASP, and PCI related to security requirements. This crafted standard document is the basis for all completeness and ambiguity analysis performed on software security requirements documents.

Thirdly, an algorithm was created to decide the final complete, ambiguous, and missing classifications concerning the completeness of the requirements document with respect to the standard document. This was performed during the post-processing of the NLP and ML resulting data through a higher level of semantic interpretation.

1.2 Outline

In chapter two, related work within similar fields and approaches is described. In chapter three, the methodology used to solve the challenge of analyzing software requirements is explained. In chapter four, the results of using different algorithms and approaches are discussed, and finally, in the last chapter, the conclusion and future work can be found.

CHAPTER 2. LITERATURE REVIEW

2.1 Related Work

Automation of requirements engineering has been used in many applications, such as within industrial software system, aerospace systems, and embedded systems. An adequate software requirements analysis from the inception of any project is performed by those with a vested interest (e.g. project manager, developer, testing engineer, etc.) which include requirements definition, specification, architecture, design and synthesis of software requirements for the development projects. Wilson et al. (1997) have created an automated tool, called ARM, which specifically searches a software related document based on quality indicators such as weak phrases. The reports produced by ARM are used to identify specification statements and structural areas of the requirements specification document. The tool does not attempt to assess the correctness of the requirements specified, instead it assesses the structure of the requirements document and individual specification statements. Particularly, the vocabulary and vernacular used to state the requirements is assessed by ARM. Gnesi et al. (2005) have developed a tool for the lexical and syntactic analyses of requirements documents and have named it as the Quality Analyzer for Requirements Specification (QuARS). It detects potential linguistic defects that can cause ambiguity in the later

phases of software development. Since QuARS is limited to the defect identification and readability analysis, further amendments were made to the QuARS and an enhanced version 'QuARS Express' was released in one of the research studies performed by Bucchiarone et al. (2008). Many studies were found on classifying non-functional requirements using information retrieval techniques. Cleland-Huang et al. (2007; 2006) present a technique for automatically classifying non-functional requirements that are related to attributes such as performance, usability, scalability, and security. The results are evaluated on 30 requirements specifications developed by M.S. students as part of their term projects. The approach is also validated on an industrial data set. The outcome of this research is simply separating each software requirement into one of twelve given non-functional requirements categories. Similarly, several researchers have described automated classification approaches for predicting categories of software requirements (Casamayor, Godoy, & Campo, 2010; Doerr, Kerkow, Koenig, Olsson, & Suzuki, 2005; Kassab, Daneva, & and Ormandjieva, 2007). In (Takahashi, et al., 2014), the authors provide the classification of security requirements based on multiple dimensions such as function and risk. The purpose of the study is to help users to identify and select the desired security requirements.

Casamayor et al. (2010) created a semi-supervised learning approach for the identification of non-functional requirements and exploited the much needed feedback from users to enhance the performance of the classifier, which is primarily based on a reduced set of categorized requirements. Doerr et al. (2005) created an experience-based systematic method to analyze non-functional requirements in order to capture

the important quality aspects and further used them as guidance during the requirements elicitation process. In one of the studies conducted by Kassab et al. (2007), an attempt was made to reduce the amount of uncertainty involved in non-functional requirements.

MacDonell et al. (2005) also state the fact that since a systems analyst of specification documents or customer requirements can be limited to his or her own knowledge, certain aspects can be missed. Formal language can help to remove some elements of ambiguity from the process since they use explicit syntax and semantics that define a set of relations and objects. Therefore, they introduced a prototype toolset that assists the systems analyst or the software engineer to select and verify terms relevant to a project. The architecture of this autonomous requirements specification processing system consists of NLP tools and a term management system.

The NLP tools are responsible for tokenizing and parsing each sentence to extract all unique nouns. The term management system filters out unimportant terms. It then classifies the remaining terms into functional categories, entities, or attributes, and inserts the objects of interest into a project knowledgebase.

There are three advantages to this system. Firstly, it checks the syntax so that it finds certain grammar errors. Secondly, it contributes to the knowledgebase by adding some keywords. Thirdly, it connects the requirements phase to the design phase, so that in the latter phase some requirements will not be missed.

There are four limitations to this system. Firstly, it is unable to disambiguate syntactic parse trees, and it does not consider some of the nested terms. Secondly, it lacks

semantic analysis. Thirdly, it is semi-automatic, since it relies on a human to decide on useful words. Fourthly, it is limited to only translating key words of requirements' nouns to a function, entity, or attribute in the knowledgebase.

Only one study performed by Takahashi et al. (2014) helped the users to identify and select the desired security dimensions and additionally provides classification of security requirements. Their approach attempts to maintain a balance between security and usability by suggesting different security requirements to the user and generating the software requirements based on the user's selections. While convenient for users, this design confines them to a limited questionnaire of suggested and available features, rather than having the freedom to choose features outside of that list.

This literature review of related work demonstrates a range of approaches to the aim of work as stated in this thesis. Some approaches use NLP, while others do not. Certain approaches only classify the security related features within a software requirements document but do nothing further. Yet others identify the completeness and ambiguity of the requirements document, similar to the work in this thesis. However, none of the approaches are making comparisons to a standard, e.g., for analyzing the completeness or ambiguity of the requirements document. Furthermore, this thesis provides a general architecture that can be expanded for both non-functional requirements (from security to usability or maintainability) as well as operators (from completeness and ambiguity to contradictions and inconsistency). In this thesis, we are advocating a semi-automatic method that can assess the completeness, incompleteness, and ambiguity in

the security requirements of software systems. This unique methodology will be discussed in the next section.

CHAPTER 3. METHODOLOGY

3.1 Approach

In order to analyze software requirements for their completeness and ambiguity, NLP, ML, and Neural Network techniques have been utilized.

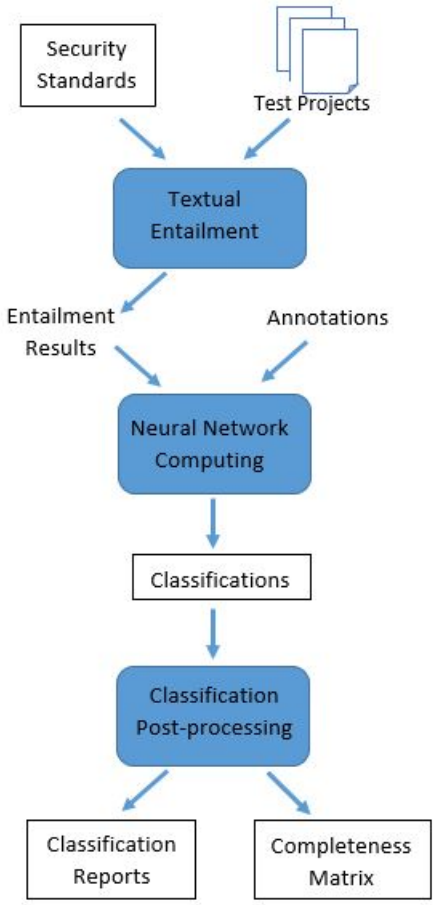

Figure 3.1 End-to-End Process in Analyzing the Security Features

Instead of creating a monolithic system that is difficult to develop and maintain, this thesis takes a modular approach by building a system composed of many components. As shown in Figure 3.1, the output of one component becomes the input of the next, making this approach flexible. Component modification and replacement is easily achieved, giving reusability of each component for a different project or domain.

The approach used in this thesis is divided into three tasks: i) collecting a list of security standards, ii) processing the text of the standards and test requirements documents, and iii) defining the percentage of completeness or ambiguity of these test documents with respect to these standards. Below we describe all these tasks in detail.

3.1.1 Identifying a list of prevalent security standards

Different organizations have always attempted to create a catalog for various software standardizations, including security features. A few of these available catalogs have been analyzed to be used as a base to create gold standards (standard security requirements) for this thesis. The catalog used in this thesis are:

- International Organization for Standardization (ISO/IEC 27001:2005, ISO/IEC 13335-1:2004, ISO/IEC 15408-1:2009). Each catalog contains more than sixty pages which are designed specifically with the purpose of evaluating security properties of IT products (Standards, 2015; ISO, 2015). The security specifications for software applications introduced in these catalogs include features such as authentication, authorization, access control, data integrity, and encryption.

- OWASP, the Open Web Application Security Project, is also a valuable resource which suggests different security features and the levels of security that should be considered in software development (OWASP, 2015). For example, the OWASP application security requirements document draft is a source defining the authentication and authorization requirements necessary for secure software systems (Fisher, 2007).
- The Payment Card Industry (PCI) data security standard (PCI Security Standards Council LLC, 2010) and other publicly available data sets.

Each of these resources contain a list of standards that organizations should establish, implement, and maintain in order to secure their physical and intellectual properties. In this thesis, all of these resources have been reviewed and a sample set relevant to the software development security specification has been extracted.

As mentioned above, the standard data from which all the security standards have been formed have been collected from ISO, OWASP, and PCI. Four examples of such standard statements include (Standards, 2015):

1) “There shall be a formal user registration and de-registration procedure in place for granting and revoking access to all information systems and services.”

2) “The allocation and use of privileges shall be restricted and controlled.”

3) “The allocation of passwords shall be controlled through a formal management process.”

4) “Management shall review users’ access rights at regular intervals using a formal process.”

For the purpose of this thesis, software requirements documents of stakeholders are required. The 15 software requirements documents associated with the 15 software projects used in study of Cleland-Huang et al. (Automated Detection and Classification of Quality Requirements, August, 2007) have been used in this thesis as test documents. Prior to their use in this thesis, the sentences containing security-related features were extracted, reducing the overall size of total sentences for each project document. These security features are then analyzed and evaluated against the standards.

Among these 15 projects, project 9 and project 11 do not contain any security related requirements for further analysis.

As an example, the following are the three test statements for project two, which is one of the above mentioned 15 projects (Cleland-Huang, Settimi, Xuchang, & Solc, 2006; Mohamed Farid, 2011):

1) “Only registered realtors shall be able to access the system.”

2) “Every user of the system shall be authenticated and authorized.”

3)”The product shall prevent its data from incorrect data being introduced.”

3.1.2 Text Processing Module

Since both the standards collected and the requirements documents obtained from stakeholders are free flowing text, Natural Language Processing (NLP) techniques have been used to first parse the text, and then Machine Learning-based entailment

algorithms have been used to compare each test document against the standards, identifying the relationships, i.e., entailment or non-entailment, between each standard statement and each test document statement. According to the PASCAL Recognizing Textual Entailment Challenge (Giampiccolo, Magnini, & Szpektor, 2006), textual entailment is defined as the one-way relationship between two statements. Given two text fragments, text (T) and hypothesis (H), entailment occurs when the meaning of H can be inferred from T (T entails H). For example, the standard statement (T), "There shall be a formal user registration and de-registration procedure in place for granting and revoking access to all information systems and services." entails the first statement in the test document (H) for project two, "Only registered realtors shall be able to access the system."

This thesis mainly adopts, modifies, and leverages the open source Excitement Open Platform (EOP) to determine whether two statements are semantically similar (Magnini, et al., 2014).

The Excitement Open Platform (EOP) is a generic architecture for textual inference. It consists of the two separate modules of Linguistic Analysis Pipeline (LAP) and the Entailment Core (EC). This platform also includes knowledge resources containing lexical and syntactic resources. The input of EOP is a pair of text and hypothesis and the output is an entailment decision and a confidence score. The two main components of EOP which make up the NLP module are: Linguistic Analysis Pipeline (LAP) and Entailment Decision Algorithms (EDAs).

This thesis mainly adopts, modifies, and leverages the open source Excitement Open Platform (EOP) to determine whether two statements are semantically similar (Magnini, et al., 2014).

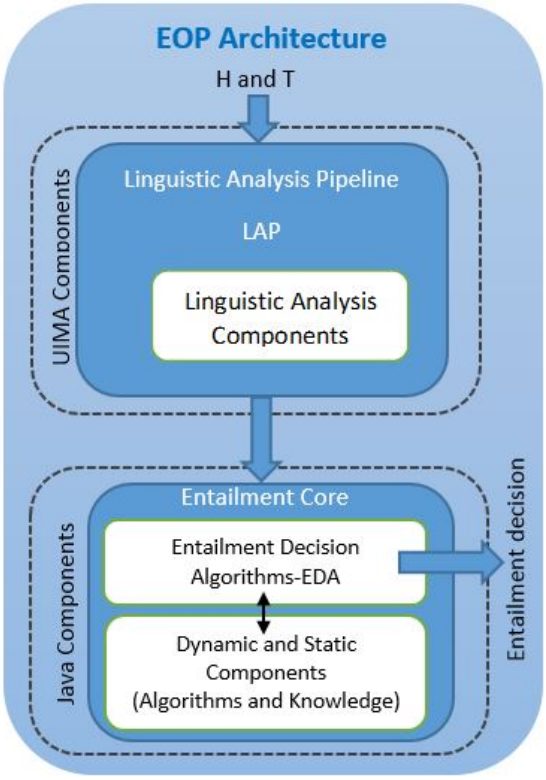

Figure 3.2 EOP Architecture

The Excitement Open Platform (EOP) is a generic architecture for textual inference. It consists of the two separate modules of Linguistic Analysis Pipeline (LAP) and the Entailment Core (EC). This platform also includes knowledge resources containing lexical and syntactic resources. The input of EOP is a pair of text and hypothesis and the output is an entailment decision and a confidence score. The two main components of EOP

which make up the NLP module are: Linguistic Analysis Pipeline (LAP) and Entailment Decision Algorithms (EDAs).

3.1.2.1 LAP

LAP, which is responsible for linguistic annotations, is a collection of annotation components for NLP, which can range from tokenization to part-of-speech tagging, chunking, named entity recognition, and parsing.

Three types of LAP that are being used are:

1. Malt Parser
2. OpenNLP Tagger
3. Tree Tagger

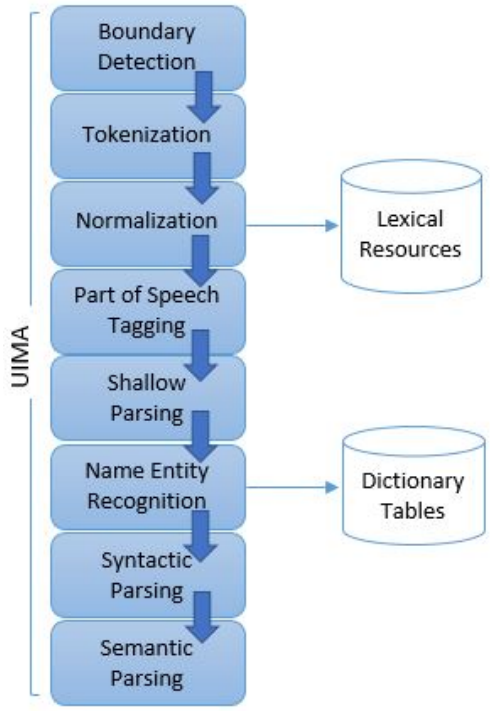

Figure 3.3 Linguistics analyzing pipeline

3.1.2.1.1 Malt Parser

Malt Parser is a data-driven parser used for unstructured text. It generates a parser from the given data using treebank data. Malt Parser uses the following nine deterministic parsing algorithms to build labeled dependency graphs: Nivre arc-eager, Nivre arc-standard, Covington non-projective, Covington projective, Stack projective, Stack swap-eager, Stack swap-lazy, Planar (implemented by Carlos Gómez-Rodríguez), 2-planar (implemented by Carlos Gómez-Rodríguez) (Nivre, 2008). It learns from the historic data and determines the next parser action (Nivre, et al., 2007).

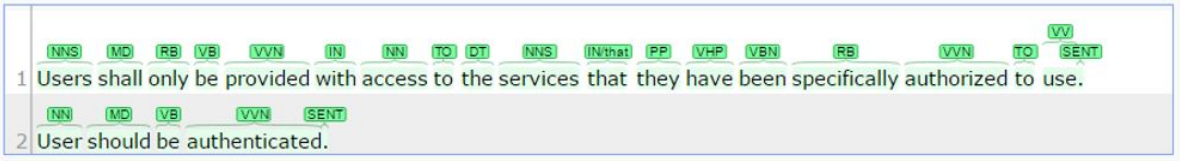

Figure 3.4 Part-Of-Speech Tagging

3.1.2.1.2 OpenNLP

OpenNLP is an ML-based tool which performs NLP tasks (tokenization, sentence segmentation, part-of-speech tagging, named entity recognition, chunking, parsing, and co-reference resolution) on unstructured data (The Apache Software Foundation, 2016).

3.1.2.1.3 TreeTagger

TreeTagger is a language independent tool developed by Helmut Schmid (1994) for annotating text with part-of-speech and lemma information in order to perform linguistic pipeline processing.

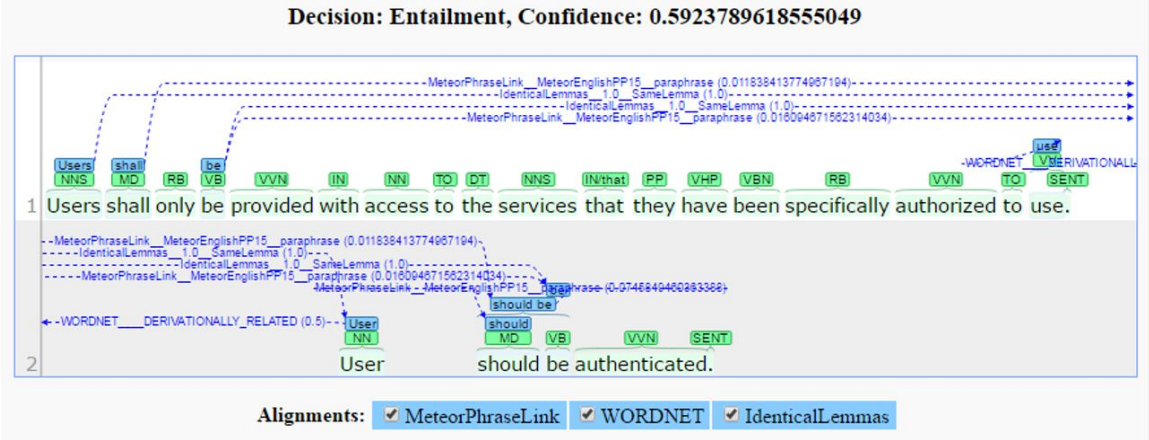

Figure 3.5 Entailment visualization of an H and T pair

3.1.2.2 EDA

Entailment Core (EC) consists of one or more Entailment Decision Algorithms (EDA), such as the transformation-based EDA, edit distance EDAs, and classification-based EDAs (Magnini, et al., 2014). It also contains zero or more subroutine components, such as scoring, annotation, lexical knowledge, and syntactic knowledge components.

After linguistic, syntactic, and semantic analysis, three different algorithms are applied for the purpose of additional semantic analysis at the statement level to determine whether the test statement can be inferred from the standard statement.

The three main available entailment algorithms are the following:

1. Tree Edit Distance algorithm
2. Edit Distance algorithm with Particle Swarm Optimization (PSO)
3. Maximum Entropy Classification entailment decision algorithm

3.1.2.2.1 Tree Edit Distance Algorithm

In order to perform textual entailment using the Tree Edit Distance algorithm, (Kouylekov & Magnini, 2005) a dependency tree of T is mapped to a dependency tree of H. This mapping consists of the three operations insertion, deletion, and substitution, each associated with a certain cost. Based on the cost of these mapping operations, which can be a combination of the cost of insertion, deletion, or substitution, the decision for entailment or non-entailment is made.

The entailment score of a given pair can be calculated as follows (Kouylekov & Magnini, 2005):

$$Score(T, H) = \frac{ed(T, H)}{ed(, H)}$$

In this formula, ed (T, H) calculates the edit distance cost and ed (,H) provides the cost of inserting the tree H.

3.1.2.2.2 Edit Distance Algorithm with Particle Swarm Optimization (PSO)

Similar to the previous algorithm, the Edit Distance PSO algorithm (Mehdad & Magnini, 2009) maps the dependency tree of T to the dependency tree of H using the three operations of insertion, deletion, and substitution, each associated with a certain cost. In contrast to the Tree Edit Distance algorithm, this algorithm optimizes these three

operations. Based on the data in the training set, this algorithm generates a distance model consisting of a distance threshold, then updates the configuration files with those heuristics. In the testing phase, it first calculates the total cost of distance operations. If the calculated number is less than the initially identified threshold, those two statement pairs are classified as entailment, and if it is higher than that threshold, it is classified as non-entailment. Two components can be used with this algorithm: Fixed Weight Token Edit Distance or Fixed Weight Lemma Edit Distance (Zanoli, 2015).

3.1.2.2.3 Maximum Entropy Classification Algorithm

The Maximum Entropy Classification algorithm (Nigam, Lafferty, & McCallum, 1999) is based on the general maximum entropy principle. Entropy is a measure of uncertainty in the data.

The formal definition of entropy is:

If X is a discrete variable, X€x, X~P

The entropy of X is H(X)=-ΣP(X)logP(X)

When there is no information available about the data (equally predictable outcome), the distribution of data is uniform and the entropy is maximum. Having less entropy means that the system produces more contextual information, e.g., to be used as weights for predication.

Labeled training data provides more insight, giving a set of constraints defining the class distribution, which is no longer uniform. The improved iterative scaling algorithm finds

the maximum entropy distribution based on the defined constraints (Nigam, Lafferty, & McCallum, 1999).

3.1.3 Classifying Operators Using a Neural Network

Once all of the data is annotated, a neural network model will be constructed, trained, evaluated, and utilized for the given dataset for operator prediction. Operators will be defined and discussed further in this section.

Neural network computing is analogous with how the human brain and nervous system work. It is a series of nodes that are connected and are responsible for calculating and making decisions. It consists of two main layers, input and output, with varied numbers of hidden layers in between. Nodes in different layers are connected with different weights. The way these nodes learn is based on backpropagation.

The steps of backpropagation are following:

1- First, random connection weights are initialized on the connection lines, and then for a set of inputs, the desired outputs are being defined.

2- The network calculates the output based on the given random weight.

3- The difference between the desired and calculated output are measured, which is referred to as network error. Based on that value, the connection weight is adjusted. The new weights are being calculated based on the old weight, the node's input value, the network error, and the learning rate.

Error in the nodes is being calculated and pushed back to the previous nodes. The node with the highest error gets the most adjustment (Russell & Norvig, 1995)

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$$

$$\Delta_i = (T_i - O_i) \times g'(\sum_j W_{j,i} a_j)$$

4- After all nodes are calculated, step 3 is continuously repeated with the new calculated weights and the original input until the result is closer to the desired output. Keras (Chollet, 2015) is a deep neural network learning library written in Python and has been utilized here to create a predictive neural network model. The basic model is based on the sequential model composed of layers, which has been used in this thesis. Keras is highly modular and easy to expand. It is running on top of TensorFlow or Theano. In this thesis, Keras is deployed on top of Theano (Chollet, 2015).

3.1.3.1 Defining Operators

Requirements documents are analyzed according to three operators: complete, ambiguous, and missing. These operators reflect the semantic relationships between the standard and test document statements, derived from the text processing and neural network stages.

3.1.3.1.1 Complete

“Property where all necessary parts of an entity have been provided and all relevant information is covered, at such a level of detail that no further explanation is required at that level of abstraction” (ISO, 2009).

3.1.3.1.2 Ambiguous

A requirement is ambiguous if it has multiple interpretations despite the reader's knowledge of the requirement engineering context (Kamsties, Berry, & Paech, 2001).

3.1.3.1.3 Missing

A requirement is missing if it fails to match both complete and ambiguous, meaning that there is no direct relationship between entities.

Evaluating the standard statement, "There shall be a formal user registration and de-registration procedure in place for granting and revoking access to all information systems and services." against the test document for project two containing three test statements should produce the operator "complete", since the aforementioned standard statement semantically matches the first test document statement, "Only registered realtors shall be able to access the system." The following standard statement should produce the operator "missing", "Management shall review users' access rights at regular intervals using a formal process.", since it doesn't semantically match any of the three statements in the test document (Cleland-Huang, Settimi, Xuchang, & Solc, 2006) :

1) "Only registered realtors shall be able to access the system."

2) "Every user of the system shall be authenticated and authorized."

3) "The product shall prevent its data from incorrect data being introduced."

These three sections (3.1.1, 3.1.2, 3.1.3) have described the proposed approach, namely starting with a defined set of security standards that will be evaluated against the

crafted test documents through text processing (entailment) and operator classification (neural network) in order to arrive at the determination of how complete a given test document is with respect to the given standards. Next, the implementation of this approach will be explained.

3.2 Implementation

3.2.1 Textual Entailment Implementation

In order to evaluate the entailment relationships between the statements in the standard document (T) and the statements in the test documents (H), a Java application (referred to as SRA, Security Requirements Analysis, herein) was developed on top of the EOP API. SRA implements its own modular framework to pre-process the inputs prior to EOP processing, run EOP processing in serial or parallel, and post-process the results into formatted report files.

3.2.1.1 Pre-Processing

The main task of pre-processing is transforming the standard and test document into individual transactions, where each transaction is composed of two statements (one from the standard document and one from the test document) and the entailment configuration (one of nine pre-defined packages in SRA built on top of the built-in configurations in EOP which can be extended) with which the two statements will be evaluated for an entailment relationship. Every statement within the standard will be evaluated against every statement within the test document. The combined standard document (e.g., ISO and OWASP) is composed of 239 statements, and the combined test

document (i.e., extracted security statements across 13 different customer requirements documents) is composed of 81 statements. When these documents are compared for entailment relationships, there will be 19,359 (239 * 81) unique statement pair transactions, and when each of these is evaluated using each pre-defined entailment configuration, nine total, there will be a total of 174,231 (239 * 81 * 9) total transactions.

3.2.1.2 EOP processing

Once the transaction is prepared, it can be processed in serial or parallel with other transactions (the thread pool dynamically expanding based on the number of available CPU cores), the preference in practice being for the latter. The transactions are formatted for consumption by the EOP engine, resulting in an entailment decision (i.e., Entailment, or NonEntailment) and associated confidence.

3.2.1.3 Post-Processing

The entailment decision and confidence results from each transaction are collected along with other data about the transaction, such as the statements involved, entailment configuration used, processing type (e.g., parallel), and the time duration of the comparison. All of this collected data is then formatted into CSV reports. Each report contains the transactions of one (1) standard statement against all other test document statements (81 total) for all nine (9) entailment configurations, resulting in a combined total of 729 transactions per report (1 * 81 * 9).

Table 3.1 describes all nine entailment configurations used. Most entailment configurations differentiate themselves from the others based on different sets of features (e.g., EDA, LAP, or components), however, all nine contain their own trained models created during the training phase of the learner and applied during the testing phase.

Table 3.1 Configurations Utilized

	LAP	EDA	Component
Configuration 1	Open NLP Tagger	Max ENT classification	
Configuration 2	Malt Parser	Max ENT classification	Bag of Lexes Scoring: Verb Ocean Lexicon Resource Verb Ocean Lexical Resource
Configuration 3	Malt Parser	Max ENT classification	Bag of Lexes Scoring: Wordnet Lexical Resource
Configuration 4	Tree Tagger	Max ENT classification	Bag of Lexes Scoring: Wordnet Lexical Resource Verb Ocean Lexical Resource
Configuration 5	Malt Parser	Max ENT classification	Bag of Lexes Scoring: Wordnet Lexical Resource Verb Ocean Lexical Resource Different Model file than number 6,7
Configuration 6	Malt Parser	Max ENT classification	Bag of Lexes Scoring: Wordnet Lexical Resource Verb Ocean Lexical Resource Different Model file than number 5,7
Configuration 7	Malt Parser	Max ENT classification	Bag of Lexes Scoring: Wordnet Lexical Resource Verb Ocean Lexical Resource Different Model file than number 5, 6
Configuration 8	Open NLP Tagger	Edit Distance	Fixed Weight Token Edit Distance Model based on threshold.
Configuration 9	Open NLP Tagger	Edit Distance PSO	Fixed Weight Token Edit Distance

3.2.1.4 Annotations

There is an additional empty field created for manually annotating each of the 81 entailment transactions (statement pairs), which can be one of three operators: "c" for

complete, "a" for ambiguous, or "n" for none. In total, 19 reports were annotated, representing 1,539 entailment transaction pairs (81 * 19). These annotations will be used during the neural network model training phase to create an operator classifier to predict whether the entailment results for a statement pair signify a "complete", "ambiguous", or "none" match, with respect to semantic meaning.

3.2.2 Neural Network Implementation

3.2.2.1 Necessity of Pattern Detection

Even when the entailment decision is not accurate in terms of the statements having the same semantic meaning, which is the case a majority of the time across all entailment configurations, there may be patterns in the data that can be identified and used to correctly predict the semantic relationship between statements. For example, most of the Maximum Entropy Classification configurations result in an entailment decision of Entailment when most of them are not semantically similar. The statements that are more similar will have higher confidences, whereas the ones that are not semantically similar will have lower confidences. These patterns, among others, can be found and utilized through machine learning.

In order to predict whether a standard statement matches a test document statement semantically, an operator classifier can be built, in this case, using a neural network (NN) through the Keras API within Python. The classifier is trained on specific data from the entailment reports in order to predict whether the entailment transaction results between two statements (one standard and one test) signifies that the match is

"complete", "ambiguous", or "none". In order to accomplish this, i) the data (entailment results and operator annotations) in the entailment reports needs to be extracted, formatted, and combined selectively; ii) the NN model needs to be carefully designed for each layer; and lastly, iii) the model needs to be trained on this extracted data. Then, the model can be used as a classifier to predict operators.

3.2.2.1.1 Pre-Processing

For the first step, each annotated entailment report is read in and the key information is extracted, formatted, and combined to form three types of files: data, target, and target int. The data file consists of the entailment configurations used and entailment decision and confidence results. The target file contains the operator annotations, represented in binary: [0,0,1] for "complete"; [0,1,0] for "ambiguous"; and [1,0,0] for "none". The target int file is the integer representation of the operator annotations, i.e., 2 for "complete", 1 for "ambiguous", and 0 for "none", where, for example, 2 as "complete" represents index 2 of the zero-based index binary representation in the target file of [0,0,1]. The data file is the input to the NN, whereas the target file indicates to the NN what the output should be, i.e., the classification (e.g., "complete" which is represented as [0,0,1]). The target int file is used in a later step during model prediction evaluation and will be further discussed at that point. Each of these three files are generated on a per entailment configuration and per test document basis, e.g., entailment configuration 1 and the test document associated with project 1. In all, there are 130

sets of these generated files, since there are 10 entailment configuration options (1-9, and all combined) and 13 test documents.

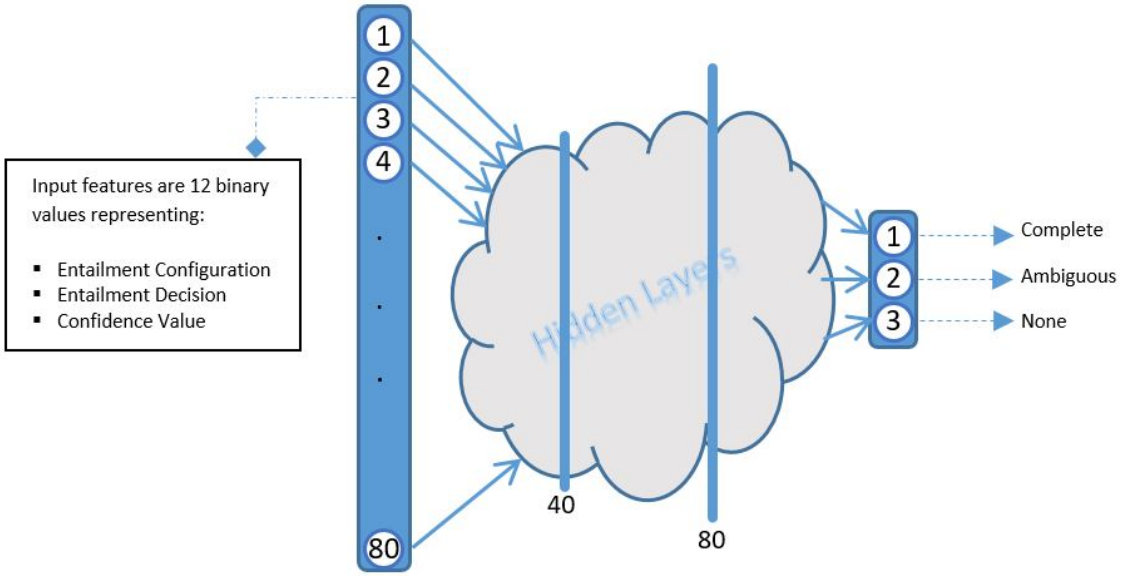

Figure 3.6 Neural Network Project Set up

3.2.2.1.2 Model Creation

During the second step, a sequential model is formed, layer by layer. Each layer is Dense (2-dimensional) and describes the number of hidden units called neurons, random weight initialization, and the activation function. The first layer has the additional descriptor of the number of input features (vector dimension), which is represented here by the data file containing the entailment configurations (9 values), decisions (2 values), and confidences (1 value), resulting in a total of 12 input features per entailment transaction.

The output of one layer feeds into the input of the next, until the last layer is reached, which is where the classification takes place. In this research, the output of the last layer can be one of three representations: 2 for "complete", 1 for "ambiguous", or 0 for "none". The model being used is shown in Figure 3.7.

```
240     model.add(Dense(80, input_dim=12, init='he_normal', activation='relu'))
241     model.add(Dense(40, init='he_normal', activation='relu'))
242     model.add(Dense(80, init='he_normal', activation='relu'))
243     model.add(Dense(3,  init='he_normal', activation='softmax'))
```

Figure 3.7 Model layers with 12 input features and 3 output classes

As shown, the input layer contains 80 neurons, the two hidden layers 40 and 80, and the last layer as the output of 3. The weight initialization selected is 'he_normal', which is a Gaussian-based initializer. The input and hidden layers all use the rectified linear unit (ReLU) activation function, whereas only the output layer uses softmax, which is ideal for this multiclass classification problems.

These parameters were carefully selected after reviewing the results of a series of tests involving 19 standard statements against each project test document for entailment configuration 1. It iterated through every combination of the available initializations, activations, and optimizers, the latter of which will be used in the next section during model compilation. Please refer to Table 3.2 for a list of these available options. Each of the 448 tests evaluated 1,539 statement pairs, comparing the predicted classification (representing either "complete", "ambiguous", or "none") with the actual classification as noted by the annotated data. There were five model combinations that performed the best, each resulting in an average precision of 73%, recall of 79%, and F-score of

0.73 with an overall accuracy of 78%. The best layer initializers were related to normal and uniform weights, which distribute the initial state best in this data through Gaussian and uniform means. The activation function of the last layer is dependent on the type of problem being solved, which in this case is a multi-class problem. Softmax is often the best choice here due to how it ensures the output layer to be properly formatted

Table 3.2 Layer activations, initializations, optimizers

Initializations (8) for 2D layers	Activations (8)	Optimizers (7)
uniform	softmax	SGD
lecun_uniform	softplus	RMSprop
normal	softsign	Adagrad
zero	relu	Adadelta
glorot_normal	tanh	Adam
glorot_uniform	sigmoid	Adamax
he_normal	hard_sigmoid	Nadam
he_uniform	linear	

Table 3.3 Top 5 Models During Model Selection

Model Components Combination	Layer Initialization	Last layer Activation	Compilation Optimizer	Accuracy	Precision	Recall	F-score
1	he_normal	Softmax	Adamax	78%	73%	79%	0.73
2	glorot_ normal	Softmax	Adam	78%	73%	79%	0.73
3	Normal	Softmax	Nadam	78%	73%	79%	0.73
4	he_uniform	Softplus	Adamax	78%	73%	79%	0.73
5	glorot_ uniform	Sigmoid	Adamax	78%	73%	79%	0.73

(i.e., all three class probabilities must add up to 1), enabling a clear distinction between the best operator match (e.g., complete) and the inferior matches for model prediction classification. These models are shown in Table 3.3. Since all 5 top models have the

same F-score, model combination 1 was arbitrarily selected (he_normal, softmax, Adamax) and used for model training.

3.2.2.1.3 Model Training

In the third step, the model formed in step two is trained on the data formatted in step one in order to build a classifier that can reliably predict the relationship between two statements (e.g., "complete", they are semantically the same) based on their corresponding entailment transaction (entailment configuration, decision, and confidence). First, the training process has to be configured by compiling the model with a loss function, optimizer, and the metrics to observe (see Figure 3.8).

```
264    model.compile(loss='categorical_crossentropy', optimizer='adamax', metrics=['accuracy'])
```

Figure 3.8 Model Compilation

Since there are three possible classifications to predict, this is a multi-classification problem, and hence, the loss function should be 'categorical_crossentropy'. As discussed previously, the Adamax optimizer was chosen during model selection based on a series of tests. Other comparable optimizers are Adam and Nadam. The metric should be set to 'accuracy' since this is a classification problem.

```
fileName='/model_weights_entailment-config_1_project_1.hdf5'
checkpointer = ModelCheckpoint(filepath=fileName,
                               monitor='val_acc',
                               save_best_only=True)
model.fit(data, target,
          verbose=False,
          batch_size=300,
          nb_epoch=50000,
          validation_split=.2,
          callbacks=[checkpointer])
```

Figure 3.9 Model Training

Then, the training process can begin with the following parameters: data, target, batch size, number of epochs, validation split, and optionally callbacks (see Figure 3.9). The data and target files previously created in step one are read into memory and used as the first two parameters of the model fitting. The next two parameters describe the number of samples (batch size) of the data to process at a time as well as how many times to iterate (epochs) over all the data. In this research, a batch size of 300 with 50,000 epochs was found to produce good results, which means that the data will be processed 300 entries (entailment configuration, decision, and confidence) at a time until it reaches the end of the data, and that whole process will repeat 50,000 times for model training. Due to the high processing demands, parallel processing was utilized within the Theano dependency of Keras, reducing the time to train in half. The validation split is set to 0.2 (80% set aside as training data and 20% as validation data). Callbacks allow custom processing to occur at certain stages during the training process, e.g., to get statistics and other state information during each stage. In this case, the ModelCheckpoint function was passed as a callback to allow the best model weights to

be saved at given stages with respect to the validation accuracy. These weights can later be reloaded, along with the model architecture that is also stored, in order to reload the trained model.

3.2.2.1.4 NN Model Predictions, Statement to Statement level

After the model is trained, it can be used to predict the operator that best describes the semantic relationship between the two given statements (i.e., "complete", "ambiguous", or "none") based on the input data of the entailment results (entailment configuration, decision, and confidence). This is done by calling the predict classes method on the model, i.e., model.predict_classes(inputData, verbose=0). For each standard statement, the semantic relationship is predicted between itself and each statement in the test document, creating a list of all relationships between every combination of each standard statement and each statement within the test document. These individual statement pair results alone are not useful, however, after they are interpreted at a higher semantic level, they can be used to evaluate how much of the standard is found within the given test document.

3.2.2.1.5 Post-Processing

In order to determine whether a given standard statement is satisfied within the test document, a set of rules need to be applied against the predicted operators for each test statement associated with the given single standard statement. The result will be a single classification indicating whether the standard statement is satisfied within the test document as a whole. As Figure 3.10 shows, in order to classify the standard

statement as "complete" overall (meaning it is found in the test document), at least one predicted "complete" must be found between the standard statement and a test document statement. In order to classify the standard statement as "ambiguous" overall, there must not be even one "complete" prediction between the standard statement and each test document statement, however, there must be at least one "ambiguous" prediction. In order to classify the standard statement as "missing" (or "none"), there must only be "none" predictions between the standard statement and each test document statement.

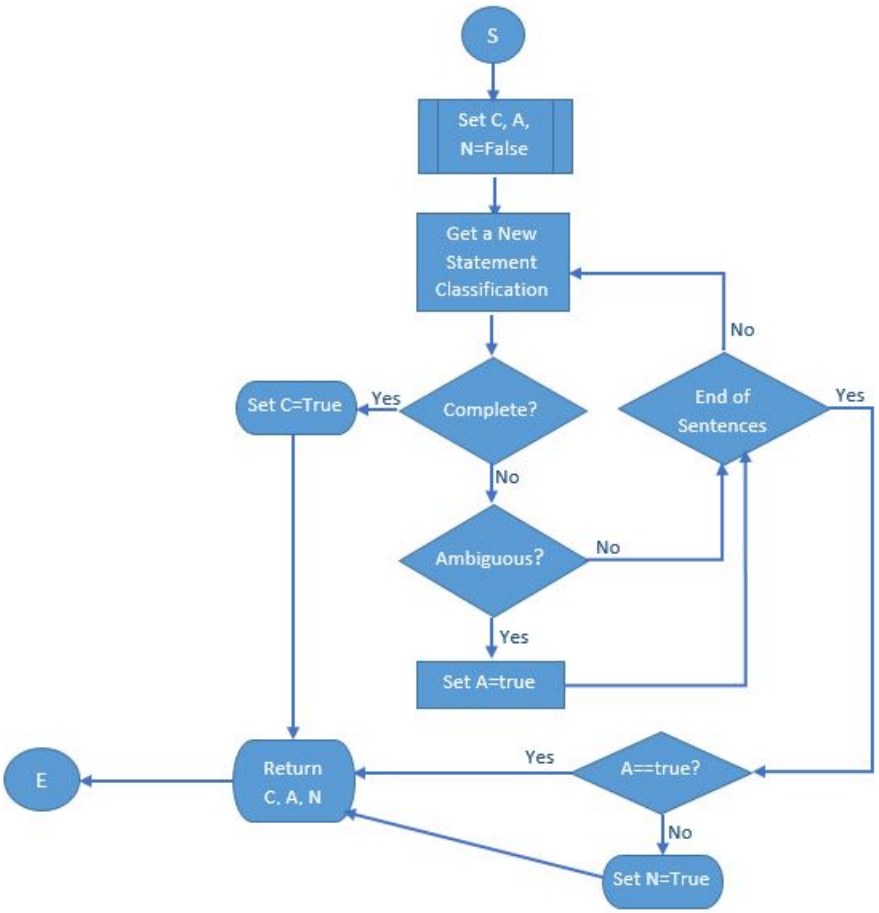

Figure 3.10 Algorithm for classifying complete, ambiguous and missing

After this process is repeated for each standard statement, a list of the predicted relationships (i.e., operators "complete", "ambiguous", or "missing") between each standard statement and the test document as a whole is obtained, e.g., "missing" signifying the standard statement was not found in the test document.

By applying the same set of rules to the manually annotated data, i.e., the operator actual values, the same type of list of relationships can be obtained between each standard statement and the test document as a whole, this time with the actual values. Taking both of these lists of classifications, predicted and actual, the following metrics can be calculated: accuracy, precision, recall, and F-score. This will be further discussed in the results section.

3.2.2.2 Model Evaluation

In order to measure the ability of the NN trained model to make correct classification predictions, the concept of the null model can be applied and evaluated against the trained models. A null model is an untrained, simple approach to prediction. Since the prediction is not correlated with the input data, it can be used to represent the worst case scenario as the performance baseline (Hooper, Couglan, & Mullen, 2008). Since there are three possible classifications to predict ("missing" as 0, "ambiguous" as 1, and "complete" as 2), three different null models are used, each representing a different potential classification. Within each null model, the same classification is hardcoded and assumed as the prediction.

For example, for the first null model (referred to with the nomenclature of "null(0)") representing the "missing" classification, it predicts that all standard statements are missing from the respective test document. The second null model, "null(1)", predicts that all standard statements are ambiguous with respect to the test document, and the third null model, "null(2)", predicts that all standard statements are complete, satisfying the standard within the test document. Table 3.2 indicates the null model results for each project test document based on the average F-score.

Table 3.2 Null Model Evaluation by Average F-score

Null Model Evaluation by average 2			
Project	**null(0)**	**null(1)**	**null(2)**
1	0.42	0.15	0.02
2	0.25	0.11	0.15
3	0.15	0.20	0.15
4	0.25	0.25	0.04
5	0.36	0.07	0.11
6	0.25	0.04	0.25
7	0.42	0.11	0.04
8	0.25	0.11	0.15
10	0.70	0.02	0.02
12	0.00	0.85	0.02
13	0.42	0.11	0.04
14	0.56	0.02	0.07
15	0.42	0.11	0.04
Average	0.34	0.17	0.08
Minimum	0.00	0.02	0.02
Maximum	0.70	0.85	0.25

As the results show, the null(0) evaluation has the highest average F-score at 0.34, followed by null(1) at nearly half the F-score, 0.17, and finally, null(2) at nearly half the

F-score of null(1), 0.08. This null model F-score distribution reveals the fact that the data is imbalanced, that is, there are far more 0 or "missing" standard statements from the test documents, followed by "ambiguous", and lastly, "complete".

Since most of the standard statements are annotated as missing from the respective test documents, overall, the null(0) evaluation makes the best assumption based on this imbalance for its predictions.

Table 3.3 Comparing the Null and NN Trained Models

NN Trained Models - comparison with the null model by average F-score						
Project	**Worst Model**	**null(0) delta**	**Best Model**	**null(0) delta**	**entailment configuration 9**	**null(0) delta**
1	0.51	0.09	0.91	0.49	0.91	0.49
2	0.37	0.12	0.73	0.48	0.73	0.48
3	0.21	0.06	0.63	0.48	0.63	0.48
4	0.35	0.10	0.72	0.47	0.49	0.24
5	0.39	0.03	0.79	0.43	0.65	0.29
6	0.37	0.12	0.69	0.44	0.57	0.32
7	0.51	0.09	0.79	0.37	0.64	0.22
8	0.33	0.08	0.68	0.43	0.52	0.27
10	0.74	0.04	0.94	0.24	0.79	0.09
12	0.79	0.79	0.94	0.94	0.79	0.79
13	0.59	0.17	0.89	0.47	0.84	0.42
14	0.64	0.08	0.84	0.28	0.84	0.28
15	0.46	0.04	0.79	0.37	0.70	0.28
average	0.48	0.14	0.80	0.45	0.70	0.36
minimum	0.21	0.03	0.63	0.24	0.49	0.09
maximum	0.79	0.79	0.94	0.94	0.91	0.79

Next, the trained NN models need to be compared against the baseline of the null model evaluation, taking the best null model results, null(0). Table 3.3 shows this

comparison for each project test document using based on the average F-score. The worst and best models, based on entailment configuration, are each compared against the best null model, and the resulting delta is shown. Note that all deltas are positive numbers, signifying that in every case, the models make better predictions in compare to the null models. On average, the worst models have an F-score 0.14 higher than this best null model, and the best models have an F-score of 0.45 higher on average. The model for entailment configuration 9 is also evaluated since it is the best model overall, and it has on average an F-score of 0.36 higher, indicating its superiority.

These results show that all models (entailment configurations 1 to 9) perform better than the best baseline null model. The justification for why entailment configuration 9 is the best will be further discussed in the results section.

3.2.3 End-to-end Demonstration of the Project Two Implementation

To illustrate the end-to-end process, project two with entailment configuration nine will be explored in detail. Within this thesis test document (Cleland-Huang, Settimi, Xuchang, & Solc, 2006) are the three following statements:

- “Only registered realtors shall be able to access the system.”
- “Every user of the system shall be authenticated and authorized.”
- “The product shall prevent its data from incorrect data being introduced.”

Using the Security Requirements Analysis Java application built during this research, these three test statements, along with a subset of security standard statements, are loaded and processed for entailment relationships. There are 19 reports generated,

each representing a standard statement against all test statements, and two of these reports are shown below in Table 3.4 and Table 3.5.

Table 3.4 Entailment Report One

Standard document (text) statement:							
User registration: There shall be a formal user registration and de-registration procedure in place for granting and revoking access to all information systems and services.							
Classification annotation	Test document (hypothesis) statement						
C	Only registered realtors shall be able to access the system.						
N	Every user of the system shall be authenticated and authorized.						
N	The product shall prevent its data from incorrect data being introduced.						
		Decision	**Confidence**	**Started on**	**Duration (sec)**	**Processing type**	
Entailment Config 1	MaxEntClassification EDA_Base+OpenNLP_EN.xml	Entailment	0.875330079	Mon 2016.07.25 10:38:31 AM EDT	7.673	PARALLEL	
	MaxEntClassification EDA_Base+ OpenNLP_EN.xml	Entailment	0.875330079	Mon 2016.07.25 10:38:31 AM EDT	7.657	PARALLEL	
	MaxEntClassification EDA_Base+ OpenNLP_EN.xml	Entailment	0.672825835	Mon 2016.07.25 10:38:39 AM EDT	5.438	PARALLEL	
Entailment Config 2	MaxEntClassification EDA_Base+VO+TP+ TPPos+TS_EN.xml	Entailment	0.884251369	Mon 2016.07.25 10:41:06 AM EDT	24.16	SERIAL	
	MaxEntClassification EDA_Base+VO+TP+ TPPos+TS_EN.xml	Entailment	0.883553955	Mon 2016.07.25 10:41:31 AM EDT	23.437	SERIAL	
	MaxEntClassification EDA_Base+VO+TP+ TPPos+TS_EN.xml	Entailment	0.673594449	Mon 2016.07.25 10:41:54 AM EDT	23.642	SERIAL	

Table 3.4 Continued

		Decision	Confidence	Started on	Duration (sec)	Processing type
Entailment Config 3	MaxEntClassification EDA_Base+WN+TP+ TPPos+TS_EN.xml	Entailment	0.8857 99892	Mon 2016.07.25 11:12:39 AM EDT	30.689	SERIAL
	MaxEntClassification EDA_Base+WN+TP+ TPPos+TS_EN.xml	Entailment	0.8848 06181	Mon 2016.07.25 11:13:09 AM EDT	30.597	SERIAL
	MaxEntClassification EDA_Base+WN+TP+ TPPos+TS_EN.xml	Entailment	0.6723 10192	Mon 2016.07.25 at 11:13:40 AM EDT	30.566	SERIAL
Entailment Config 4	MaxEntClassification EDA_Base+WN+ VO_EN.xml	Entailment	0.9384 91431	Mon 2016.07.25 11:53:24 AM EDT	18.943	PARALLEL
	MaxEntClassification EDA_Base+WN+ VO_EN.xml	Entailment	0.9384 91431	Mon 2016.07.25 11:53:24 AM EDT	18.49	PARALLEL
	MaxEntClassification EDA_Base +WN+VO_EN.xml	Entailment	0.7694 71599	Mon 2016.07.25 11:53:43 AM EDT	16.554	PARALLEL
Entailment config 5	MaxEntClassification EDA_Base+WN+VO+ TP+TPPos_EN.xml	Entailment	0.9309 32216	Mon 2016.07.25 11:59:48 AM EDT	30.218	SERIAL
	MaxEntClassification EDA_Base+WN+VO+ TP+TPPos_EN.xml	Entailment	0.9217 96657	Mon 2016.07.25 12:00:18 PM EDT	30.829	SERIAL
	MaxEntClassification EDA_Base+WN+VO+TP +TPPos_EN.xml	Entailment	0.7481 22454	Mon 2016.07.25 12:00:49 PM EDT	29.875	SERIAL
Entailment Config 6	MaxEntClassification EDA_Base+WN+VO+TP +TPPos+TS_EN.xml	Entailment	0.8929 24711	Mon 2016.07.25 12:40:41 PM EDT	30.427	SERIAL
	MaxEntClassification EDA_Base+WN+VO+ TP+TPPos+TS_EN.xml	Entailment	0.8908 97041	Mon 2016.07.25 12:41:12 PM EDT	30.464	SERIAL

Table 3.4 Continued

		Decision	Confidence	Started on	Duration (sec)	Processing type
Config 6	MaxEntClassification EDA_Base+WN+VO+ TP+TPPos+TS_EN.xml	Entailment	0.6839 44088	Mon 2016.07.25 12:41:42 PM EDT	30.179	SERIAL
Entailment Config 7	MaxEntClassification EDA_Base+WN+ VO+TS_EN.xml	Entailment	0.8993 74716	Mon 2016.07.25 01:21:43 PM EDT	30.235	SERIAL
	MaxEntClassification EDA_Base+ WN+VO+TS_EN.xml	Entailment	0.9029 70467	Mon 2016.07.25 01:22:13 PM EDT	30.314	SERIAL
	MaxEntClassification EDA_Base+WN+ VO+TS_EN.xml	Entailment	0.6959 27525	Mon 2016.07.25 01:22:43 PM EDT	30.177	SERIAL
Entailment Config 8	EditDistance EDA_EN.xml	Non Entailment	0.0508 24176	Mon 2016.07.25 02:01:36 PM EDT	7.626	PARALLEL
	EditDistance EDA_EN.xml	Entailment	0.0741 75824	Mon 2016.07.25 02:01:36 PM EDT	7.641	PARALLEL
	EditDistance EDA_EN.xml	Non Entailment	0.3147 13065	Mon 2016.07.25 02:01:43 PM EDT	5.297	PARALLEL
Entailment Config 9	EditDistancePSO EDA _EN.xml	Entailment	0.0247 71018	Mon 2016.07.25 02:03:33 PM EDT	5.312	PARALLEL
	EditDistance PSOEDA _EN.xml	Entailment	0.0893 98999	Mon 2016.07.25 02:03:33 PM EDT	5.328	PARALLEL
	EditDistance PSOEDA _EN.xml	Non Entailment	0.1232 87472	Mon 2016.07.25 02:03:38 PM EDT	4.377	PARALLEL

Table 3.5 Entailment Report Nineteen

Standard document (text) statement:						
The risks to the organization's information and information processing facilities from business processes involving external parties shall be identified and appropriate controls implemented before granting access.						
Classification Annotation	**Test document (hypothesis) statement**					
A	Only registered realtors shall be able to access the system.					
C	Every user of the system shall be authenticated and authorized.					
N	The product shall prevent its data from incorrect data being introduced.					
		Decision	**Confidence**	**Started on**	**Duration (sec)**	**Processing type**
Entailment Config 1	MaxEntClassification EDA_Base+OpenNLP _EN.xml	Entailment	0.91151	Sat 2016.07.30 03:34:38 PM EDT	5.564	PARALLEL
	MaxEntClassification EDA_Base+OpenNLP _EN.xml	Entailment	0.87533	Sat 2016.07.30 03:34:38 PM EDT	5.58	PARALLEL
	MaxEntClassification EDA_Base+OpenNLP _EN.xml	Entailment	0.80752	Sat 2016.07.30 03:34:43 PM EDT	6.594	PARALLEL
Entailment Config 2	MaxEntClassification EDA_Base+VO+TP+ TPPos+TS_EN.xml	Entailment	0.88968 1547	Sat 2016.07.30 at 03:37:24 PM EDT	22.876	SERIAL
	MaxEntClassification EDA_Base+VO+TP+ TPPos+TS_EN.xml	Entailment	0.86819 6797	Sat 2016.07.30 03:37:47 PM EDT	23.439	SERIAL
	MaxEntClassification EDA_Base+VO+TP+ TPPos+TS_EN.xml	Entailment	0.80195 1338	Sat 2016.07.30 03:38:11 PM EDT	21.829	SERIAL

Table 3.5 Continued

		Decision	Confidence	Started on	Duration (sec)	Processing type
Entailment Config 3	MaxEntClassification EDA_Base+WN+TP+ TPPos+TS_EN.xml	Entailment	0.90015 3098	Sat 2016.07.30 04:08:42 PM EDT	29.844	SERIAL
	MaxEntClassification EDA_Base+WN+TP+ TPPos+TS_EN.xml	Entailment	0.87949 6338	Sat 2016.07.30 04:09:12 PM EDT	31.094	SERIAL
	MaxEntClassification EDA_Base+WN+TP+ TPPos+TS_EN.xml	Entailment	0.80273 3658	Sat 2016.07.30 04:09:43 PM EDT	31.191	SERIAL
Entailment Config 4	MaxEntClassification EDA_Base+ WN+VO_EN.xml	Entailment	0.95265 053	Sat 2016.07.30 04:50:12 PM EDT	22.842	PARALLEL
	MaxEntClassification EDA_Base+WN+ VO_EN.xml	Entailment	0.92741 8327	Sat 2016.07.30 04:50:12 PM EDT	21.619	PARALLEL
	MaxEntClassification EDA_Base+WN +VO_EN.xml	Entailment	0.89050 0164	Sat 2016.07.30 04:50:33 PM EDT	18.566	PARALLEL
Entailment Config 5	MaxEntClassification EDA_Base+WN+VO+ TP+TPPos_EN.xml	Entailment	0.94308 0533	Sat 2016.07.30 04:57:49 PM EDT	31.596	SERIAL
	MaxEntClassification EDA_Base+WN+VO+ TP+TPPos_EN.xml	Entailment	0.93372 8804	Sat 2016.07.30 04:58:21 PM EDT	31.579	SERIAL
	MaxEntClassification EDA_Base+WN+VO+ TP+TPPos_EN.xml	Entailment	0.87806 1297	Sat 2016.07.30 04:58:52 PM EDT	31.658	SERIAL

Table 3.5 Continued

		Decision	Confidence	Started on	Duration (sec)	Processing type
Entailment Config 6	MaxEntClassification EDA_Base+WN+VO+TP+TPPos+TS_EN.xml	Entailment	0.902507066	Sat 2016.07.30 05:40:41 PM EDT	31	SERIAL
	MaxEntClassification EDA_Base+WN+VO+TP+TPPos+TS_EN.xml	Entailment	0.883135585	Sat 2016.07.30 05:41:12 PM EDT	31.267	SERIAL
	MaxEntClassification EDA_Base+WN+VO+TP+TPPos+TS_EN.xml	Entailment	0.813122023	Sat 2016.07.30 05:41:44 PM EDT	32.094	SERIAL
Entailment Config 7	MaxEntClassification EDA_Base+WN+VO+TS_EN.xml	Entailment	0.908815576	Sat 2016.07.30 06:24:19 PM EDT	33.583	SERIAL
	MaxEntClassification EDA_Base+WN+VO+TS_EN.xml	Entailment	0.885645664	Sat 2016.07.30 06:24:53 PM EDT	31.392	SERIAL
	MaxEntClassification EDA_Base+WN+VO+TS_EN.xml	Entailment	0.823579864	Sat 2016.07.30 06:25:24 PM EDT	30.454	SERIAL
Entailment Config 8	EditDistance EDA_EN.xml	Non Entailment	0.050824176	Sat 2016.07.30 07:05:43 PM EDT	8.21	PARALLEL
	EditDistance EDA_EN.xml	Non Entailment	0.092490842	Sat 2016.07.30 07:05:43 PM EDT	8.257	PARALLEL
	EditDistance EDA_EN.xml	Non Entailment	0.314713065	Sat 2016.07.30 07:05:51 PM EDT	5.25	PARALLEL

Table 3.5 Continued

		Decision	Confidence	Started on	Duration (sec)	Processing type
Entailment Config 9	EditDistancePSOEDA_EN.xml	Entailment	0.0046 33056	Sat 2016.07.30 07:07:49 PM EDT	5.954	PARALLEL
	EditDistancePSOEDA_EN.xml	Non Entailment	0.0365 37801	Sat 2016.07.30 at 7:07:49 PM EDT	6.048	PARALLEL
	EditDistancePSOEDA_EN.xml	Non Entailment	0.1352 35939	Sat 2016.07.30 at 07:07:55 PM EDT	5.172	PARALLEL

The first header, "Classification annotation", contains one of three manually entered values: 'c' for complete, 'a' for ambiguous, and 'n' for none. This is the truth statement regarding the semantic relationship between the standard statement and the test statement in the given row for the given report. The date, time, and duration were tracked per statement pair transaction for each entailment configuration, each processed according to the respective type of parallel or serial. There is a corresponding entailment decision and confidence for the respective entailment configuration.

Using Python, the report classification annotation (target classification) and entailment configuration, decision, and confidence (input data) were extracted and formatted by entailment configuration in order to be used by Keras for NN model selection, training, and evaluation. Since there are three test statements and 19 standard statements, there will be 57 statement pair relationships to represent per entailment configuration. This representation for entailment configuration 9 is shown in Table 3.5 based on the

first 10 entries out of the total 57. For the input data header, the "Statement pair" column represents the first 10 out of 57 statement comparisons, 1-9 represent

Table 3.6 Formatted Data and Target

Input Data													Target Classification		
Statement pair	1	2	3	4	5	6	7	8	9	E	N	Confidence	n	a	C
1	0	0	0	0	0	0	0	0	1	0	1	0.154143911	0	1	0
2	0	0	0	0	0	0	0	0	1	0	1	0.157211363	0	1	0
3	0	0	0	0	0	0	0	0	1	0	1	0.209575348	1	0	0
4	0	0	0	0	0	0	0	0	1	1	0	0.079863614	0	1	0
5	0	0	0	0	0	0	0	0	1	1	0	0.019611428	0	1	0
6	0	0	0	0	0	0	0	0	1	0	1	0.105512411	1	0	0
7	0	0	0	0	0	0	0	0	1	1	0	0.004633056	0	1	0
8	0	0	0	0	0	0	0	0	1	0	1	0.036537801	0	0	1
9	0	0	0	0	0	0	0	0	1	0	1	0.135235939	1	0	0
10	0	0	0	0	0	0	0	0	1	0	1	0.022651949	1	0	0

entailment configurations 1-9, E is the decision Entailment, and N is the decision NonEntailment. For the target classification header, ‘n’ is none (index 0), ‘a’ is ambiguous (index 1), and ‘c’ is complete (index 2). Looking at the first row, represented by statement pair 1, entailment 9 is used (the '9' column is set with the binary '1') and resulted in the decision NonEntailment with a confidence of 0.154143911. The target classification (manually annotated in the report) for this statement pair transaction is ambiguous (index 1 in [0,1,0]). In a perfect classification system, the complete target classification would correspond to the entailment decision of Entailment, the none target classification to NonEntailment, and the ambiguous target classification to a

mixture of both entailment decisions. Most of the time, these patterns are true in the sample set, the exception being for statement pair 8.

In order to discover existing patterns, i.e., the relationship between the entailment results (input data) and annotations (target classification), this formatted input data and target classifications are used to train a neural network to predict target classifications based on the input data. The classification prediction results can be seen in Table 3.6 all 57 statement pairs.

Table 3.7 Classification Report for Operator Predictions

Combined Report Between 19 Standard Statements and the 3 Test Statement				
	Precision	**Recall**	**F-score**	**Support**
None	93%	90%	0.92	42
Ambiguous	89%	100%	0.94	8
Complete	57%	57%	0.57	7
Avg/Total	88%	88%	0.88	57
Accuracy	87.7			

Table 3.8 Classification Report of Standard Statement One

Individual Statement Classifications				
	Precision	**Recall**	**F-score**	**Support**
None	50%	50%	0.5	2
Complete	0%	0%	0	1
Avg/Total	33%	33%	0.33	3
Accuracy	33.3			

Table 3.9 Classification Report of Statement Nineteen

Individual Statement Classifications				
	Precision	Recall	F-score	Support
None	100%	100%	1	1
Ambiguous	100%	100%	1	1
Complete	100%	100%	1	1
Avg/Total	100%	100%	1	3
Accuracy	100			

There are 42 actual none classifications, 8 ambiguous, and 7 complete. The most difficult classification to predict in this example is "complete". The neural network is able to correctly identify 57% (recall) of the actual complete relationships, and when it predicts complete, 57% (precision) of the time it's correct. Table 3.7 and Table 3.8 break down the results by standard statement, corresponding with entailment reports 1 and 19 respectively. In each, there are three statement pair comparisons, each of the three test statements against standard statement 1 from entailment report 1. Applying the

Table 3.10 Standard Statement Actual and Predicted Classifications

Standard Statement	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Report Classification Actual	2	0	0	0	0	1	2	1	0	0	2	2	2	1	0	0	2	1	1
Report Classification Predicted	2	2	1	2	0	1	0	1	0	0	2	2	0	1	0	0	2	1	1

algorithm from Figure 3.10 to each of the 19 sets of classifications represented by a single standard statement against each of the 3 test statements, a list of final

classifications can be obtained for both actual and predicted values, as shown in Table 3.9.

Table 3.10 shows the metrics calculated from the list of actual and predicted values. This classification report shows the ability to correctly classify the relationship between the standard statements and a given test document, i.e., whether the standard statements are completely satisfied by the test document, partially, or not at all, that is, missing.

Table 3.11 Overall Classification Report of Nineteen Standard Statements

Classification Report				
	Precision	**Recall**	**F-score**	**Support**
Missing	71%	62%	0.67	8
Ambiguous	83%	100%	0.91	5
Complete	67%	67%	0.67	6
Avg/Total	73%	74%	0.73	19
Accuracy	73.70%			

CHAPTER 4. RESULTS AND DISCUSSION

The methodology adopted in this research leads to two main categories of results centered around the analysis and presentation of the completeness for the given software requirements document with respect to security. The first category of results is classification reports, which demonstrate the ability of the NN model to predict whether the standard statements are found, or satisfied, within the test document for the given project using the defined operators (complete, ambiguous, and missing), per entailment configuration option. The second category of results is the completeness matrix, which asserts, through model predictions, the percentage of the given project test document that is complete, ambiguous, and missing.

4.1 Classification Report

As was mentioned in the NN post-processing section, the two lists of classifications, predicted and actual, representing the completeness of the standard statements in a single test document, can be used to calculate the following metrics: accuracy, precision, recall, and F-score. These metrics represent the results of one out of 130 possible combinations of input test documents (13 projects) and entailment configurations (1 through 9 configuration and all combined) across the static set of standard statements. Table 4.1 through Table 4.13 show the results of these 130 combinations.

Table 4.1 Project One Classification Report

Project 1				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	74%	77%	74%	0.74
2	84%	89%	84%	0.84
3	84%	89%	84%	0.85
4	79%	82%	79%	0.79
5	84%	84%	84%	0.84
6	79%	86%	79%	0.80
7	79%	86%	79%	0.80
8	63%	46%	63%	0.51
9	89%	95%	89%	0.91
all	58%	78%	58%	0.57

Table 4.2 Project Two Classification Report

Project 2				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	42%	42%	42%	0.38
2	58%	67%	58%	0.53
3	47%	48%	47%	0.44
4	53%	54%	53%	0.52
5	68%	76%	68%	0.68
6	68%	70%	68%	0.68
7	53%	54%	53%	0.50
8	47%	61%	47%	0.44
9	74%	73%	74%	0.73
all	42%	45%	42%	0.37

Table 4.3 Project Three Classification Report

Project 3				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	37%	37%	37%	0.37
2	58%	61%	58%	0.57
3	42%	44%	42%	0.42
4	53%	55%	53%	0.52
5	26%	25%	26%	0.25
6	42%	45%	42%	0.39
7	37%	37%	37%	0.36
8	37%	25%	37%	0.29
9	63%	65%	63%	0.63
all	26%	19%	26%	0.21

Table 4.4 Project Four Classification Report

Project 4				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	63%	64%	63%	0.63
2	47%	28%	47%	0.35
3	58%	73%	58%	0.54
4	74%	77%	74%	0.72
5	74%	77%	74%	0.72
6	53%	71%	53%	0.45
7	47%	48%	47%	0.45
8	53%	41%	53%	0.44
9	53%	58%	53%	0.49
all	68%	69%	68%	0.68

Table 4.5 Project Five Classification Report

Project 5				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	58%	56%	58%	0.46
2	79%	82%	79%	0.79
3	63%	53%	63%	0.57
4	63%	78%	63%	0.56
5	58%	45%	58%	0.50
6	58%	47%	58%	0.50
7	47%	35%	47%	0.39
8	58%	57%	58%	0.57
9	68%	75%	68%	0.65
all	47%	52%	47%	0.42

Table 4.6 Project Six Classification Report

Project 6				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	53%	62%	53%	0.44
2	58%	63%	58%	0.51
3	53%	67%	53%	0.52
4	58%	79%	58%	0.53
5	42%	65%	42%	0.37
6	58%	79%	58%	0.53
7	58%	54%	58%	0.50
8	53%	49%	53%	0.49
9	58%	62%	58%	0.57
all	68%	74%	68%	0.69

Table 4.7 Project Seven Classification Report

Project 7				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	74%	73%	74%	0.73
2	74%	75%	74%	0.72
3	68%	72%	68%	0.68
4	74%	77%	74%	0.72
5	63%	60%	63%	0.61
6	79%	81%	79%	0.77
7	79%	88%	79%	0.80
8	53%	53%	53%	0.53
9	63%	68%	63%	0.64
all	47%	65%	47%	0.51

Table 4.8 Project Eight Classification Report

Project 8				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	42%	29%	42%	0.33
2	53%	63%	53%	0.48
3	68%	76%	68%	0.68
4	42%	61%	42%	0.41
5	42%	46%	42%	0.42
6	58%	68%	58%	0.57
7	47%	33%	47%	0.34
8	63%	47%	63%	0.54
9	53%	53%	53%	0.52
all	58%	74%	58%	0.54

Table 4.9 Project Ten Classification Report

Project 10				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	84%	79%	84%	0.81
2	84%	94%	84%	0.86
3	84%	76%	84%	0.79
4	95%	95%	95%	0.94
5	84%	94%	84%	0.86
6	79%	82%	79%	0.80
7	95%	95%	95%	0.94
8	79%	70%	79%	0.74
9	79%	79%	79%	0.79
all	74%	92%	74%	0.78

Table 4.10 Project Twelve Classification Report

Project 12				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	89%	89%	100%	0.94
2	79%	95%	79%	0.85
3	84%	100%	82%	0.90
4	74%	88%	82%	0.85
5	74%	88%	74%	0.80
6	89%	95%	89%	0.91
7	84%	95%	84%	0.88
8	89%	89%	100%	0.94
9	74%	93%	74%	0.79
all	68%	100%	65%	0.79

Table 4.11 Project Thirteen Classification Report

Project 13				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	68%	77%	68%	0.69
2	68%	81%	68%	0.69
3	58%	66%	58%	0.59
4	84%	84%	84%	0.84
5	68%	73%	68%	0.70
6	74%	72%	74%	0.73
7	89%	91%	89%	0.89
8	79%	85%	79%	0.74
9	84%	84%	84%	0.84
all	58%	76%	58%	0.59

Table 4.12 Project Fourteen Classification Report

Project 14				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	68%	61%	68%	0.64
2	79%	82%	79%	0.79
3	84%	86%	84%	0.84
4	84%	83%	84%	0.83
5	74%	70%	74%	0.66
6	84%	86%	84%	0.84
7	74%	65%	74%	0.69
8	74%	65%	74%	0.69
9	84%	87%	84%	0.84
all	68%	82%	68%	0.70

Table 4.13 Project Fifteen Classification Report

Project 15				
Entailment Configuration	Accuracy	Precision	Recall	F-score
1	74%	75%	74%	0.74
2	63%	67%	63%	0.64
3	74%	84%	74%	0.73
4	58%	64%	58%	0.58
5	74%	74%	74%	0.74
6	58%	66%	58%	0.59
7	79%	84%	79%	0.79
8	58%	63%	58%	0.58
9	68%	75%	68%	0.70
all	47%	76%	47%	0.46

In order to determine the best entailment configuration option overall, the F-scores across each project must be averaged per each entailment configuration option. The results of this analysis is shown in Table 4.14 and Figure 4.1, which demonstrates that entailment configuration 9 has the highest average F-score, 0.70, and thus, it is the best candidate. As mentioned previously, entailment configuration 9 consists of the Open NLP tagger (LAP), Edit Distance PSO (EDA), and Fixed Weight Token Edit Distance (component).

It was unexpected to see that the entailment configuration "all" did not have the best overall F-score since it combines all of the data for entailment configurations 1-9. It appears, however, that this combination leads to confusion within the neural network, resulting in lower predictive power. Entailment configuration 9 being the best candidate for model predictions was also unexpected, given the fact that during the entailment text processing phase, nearly every statement comparison between the

standard document and the test document resulted in the classification of NonEntailment. However, as demonstrated in these results, while the entailment decision in EOP was often biased towards missing entailment relationships,

Table 4.14 Evaluation of the Best Configuration Option

Entailment Configuration	Average F-score Across All Projects
1	0.61
2	0.66
3	0.66
4	0.68
5	0.63
6	0.66
7	0.64
8	0.58
9	0.70
all	0.56

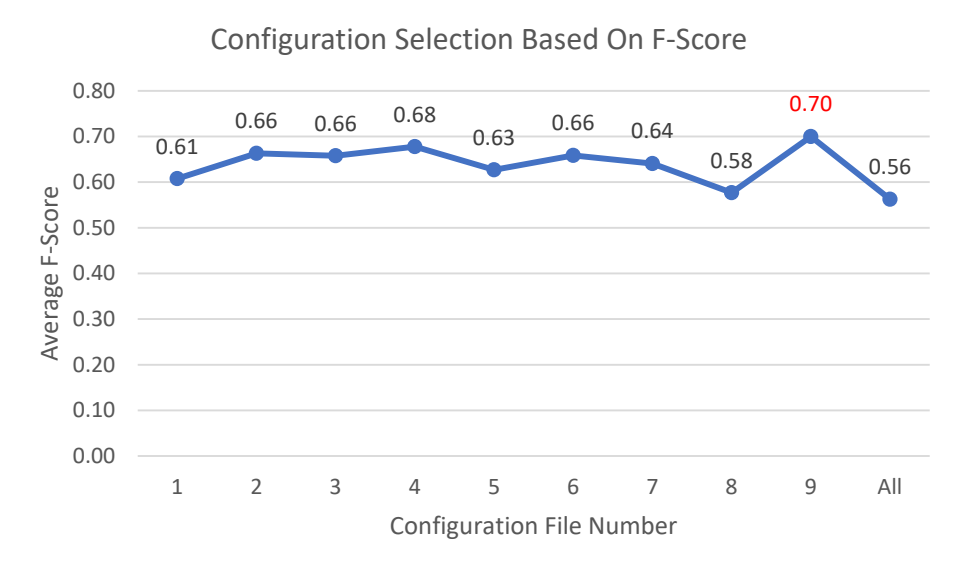

Figure 4.1 Visualization of the Best Entailment Configuration

the combination of entailment decision and confidence result in more clearly delineated patterns for the neural network to discover and utilize during model prediction.

4.2 Completeness Matrix

From the predicted classification results, within each project classification report previously demonstrated, the overall completeness of the standard being satisfied within each combination of test document and entailment configuration can be derived and represented by each operator category. Table 4.15 through Table 4.24 show these completeness results.

Table 4.15 Completeness Matrix of Configuration One

Entailment Configuration 1			
Test Document	**Complete**	**Ambiguous**	**Missing**
Project 1	15.8%	42.1%	42.1%
Project 2	15.8%	10.5%	73.7%
Project 3	26.3%	36.8%	36.8%
Project 4	15.8%	31.6%	52.6%
Project 5	5.3%	0.0%	94.7%
Project 6	10.5%	0.0%	89.5%
Project 7	10.5%	31.6%	57.9%
Project 8	0.0%	26.3%	73.7%
Project 10	21.1%	0.0%	78.9%
Project 12	0.0%	100.0%	0.0%
Project 13	15.8%	47.4%	36.8%
Project 14	21.1%	0.0%	78.9%
Project 15	21.1%	26.3%	52.6%

Table 4.16 Completeness Matrix of Configuration Two

Entailment Configuration 2			
Test Document	**Complete**	**Ambiguous**	**Missing**
Project 1	15.8%	42.1%	42.1%
Project 2	10.5%	10.5%	78.9%
Project 3	36.8%	21.1%	42.1%
Project 4	21.1%	0.0%	78.9%
Project 5	15.8%	26.3%	57.9%
Project 6	15.8%	0.0%	84.2%
Project 7	10.5%	15.8%	73.7%
Project 8	21.1%	5.3%	73.7%
Project 10	26.3%	5.3%	68.4%
Project 12	21.1%	68.4%	10.5%
Project 13	26.3%	42.1%	31.6%
Project 14	26.3%	15.8%	57.9%
Project 15	26.3%	26.3%	47.4%

Table 4.17 Completeness Matrix of Configuration Three

Entailment Configuration 3			
Test Document	**Complete**	**Ambiguous**	**Missing**
Project 1	21.1%	31.6%	47.4%
Project 2	10.5%	31.6%	57.9%
Project 3	36.8%	21.1%	42.1%
Project 4	21.1%	10.5%	68.4%
Project 5	31.6%	5.3%	63.2%
Project 6	15.8%	15.8%	68.4%
Project 7	5.3%	31.6%	63.2%
Project 8	15.8%	21.1%	63.2%
Project 10	0.0%	5.3%	94.7%
Project 12	26.3%	73.7%	0.0%
Project 13	31.6%	31.6%	36.8%
Project 14	21.1%	15.8%	63.2%
Project 15	26.3%	42.1%	31.6%

Table 4.18 Completeness Matrix of Configuration Four

Entailment Configuration 4			
Test Document	**Complete**	**Ambiguous**	**Missing**
Project 1	10.5%	42.1%	47.4%
Project 2	42.1%	21.1%	36.8%
Project 3	15.8%	47.4%	36.8%
Project 4	5.3%	47.4%	47.4%
Project 5	5.3%	5.3%	89.5%
Project 6	10.5%	5.3%	84.2%
Project 7	5.3%	31.6%	63.2%
Project 8	5.3%	52.6%	42.1%
Project 10	10.5%	5.3%	84.2%
Project 12	0.0%	84.2%	15.8%
Project 13	15.8%	21.1%	63.2%
Project 14	15.8%	10.5%	73.7%
Project 15	15.8%	47.4%	36.8%

Table 4.19 Completeness Matrix of Configuration Five

Entailment Configuration 5			
Test Document	**Complete**	**Ambiguous**	**Missing**
Project 1	10.5%	26.3%	63.2%
Project 2	15.8%	47.4%	36.8%
Project 3	47.4%	21.1%	31.6%
Project 4	5.3%	42.1%	52.6%
Project 5	21.1%	0.0%	78.9%
Project 6	5.3%	31.6%	63.2%
Project 7	26.3%	15.8%	57.9%
Project 8	21.1%	47.4%	31.6%
Project 10	26.3%	5.3%	68.4%
Project 12	10.5%	73.7%	15.8%
Project 13	21.1%	31.6%	47.4%
Project 14	5.3%	0.0%	94.7%
Project 15	15.8%	31.6%	52.6%

Table 4.20 Completeness Matrix of Configuration Six

Entailment Configuration 6			
Test Document	**Complete**	**Ambiguous**	**Missing**
Project 1	21.1%	36.8%	42.1%
Project 2	21.1%	36.8%	42.1%
Project 3	36.8%	10.5%	52.6%
Project 4	21.1%	5.3%	73.7%
Project 5	15.8%	0.0%	84.2%
Project 6	10.5%	5.3%	84.2%
Project 7	5.3%	26.3%	68.4%
Project 8	10.5%	26.3%	63.2%
Project 10	15.8%	5.3%	78.9%
Project 12	5.3%	89.5%	5.3%
Project 13	21.1%	21.1%	57.9%
Project 14	26.3%	5.3%	68.4%
Project 15	31.6%	31.6%	36.8%

Table 4.21 Completeness Matrix of Configuration Seven

Entailment Configuration 7			
Test Document	**Complete**	**Ambiguous**	**Missing**
Project 1	21.1%	36.8%	42.1%
Project 2	10.5%	42.1%	47.4%
Project 3	26.3%	31.6%	42.1%
Project 4	21.1%	21.1%	57.9%
Project 5	15.8%	0.0%	84.2%
Project 6	21.1%	0.0%	78.9%
Project 7	10.5%	47.4%	42.1%
Project 8	0.0%	10.5%	89.5%
Project 10	10.5%	5.3%	84.2%
Project 12	21.1%	73.7%	5.3%
Project 13	21.1%	15.8%	63.2%
Project 14	21.1%	0.0%	78.9%
Project 15	10.5%	42.1%	47.4%

Table 4.22 Completeness Matrix of Configuration Eight

Entailment Configuration 8			
Test Document	**Complete**	**Ambiguous**	**Missing**
Project 1	5.3%	0.0%	94.7%
Project 2	5.3%	31.6%	63.2%
Project 3	47.4%	5.3%	47.4%
Project 4	0.0%	26.3%	73.7%
Project 5	21.1%	15.8%	63.2%
Project 6	31.6%	5.3%	63.2%
Project 7	15.8%	26.3%	57.9%
Project 8	42.1%	0.0%	57.9%
Project 10	0.0%	10.5%	89.5%
Project 12	0.0%	100.0%	0.0%
Project 13	15.8%	5.3%	78.9%
Project 14	15.8%	0.0%	84.2%
Project 15	26.3%	36.8%	36.8%

Table 4.23 Completeness Matrix of Configuration Nine

Entailment Configuration 9			
Test Document	**Complete**	**Ambiguous**	**Missing**
Project 1	21.1%	31.6%	47.4%
Project 2	31.6%	31.6%	36.8%
Project 3	31.6%	47.4%	21.1%
Project 4	5.3%	21.1%	73.7%
Project 5	21.1%	5.3%	73.7%
Project 6	21.1%	21.1%	57.9%
Project 7	10.5%	36.8%	52.6%
Project 8	26.3%	21.1%	52.6%
Project 10	10.5%	10.5%	78.9%
Project 12	31.6%	63.2%	5.3%
Project 13	15.8%	21.1%	63.2%
Project 14	31.6%	5.3%	63.2%
Project 15	26.3%	31.6%	42.1%

Table 4.24 Completeness Matrix of All Configurations

Entailment Configuration all			
Test Document	**Complete**	**Ambiguous**	**Missing**
Project 1	26.3%	52.6%	21.1%
Project 2	42.1%	47.4%	10.5%
Project 3	57.9%	26.3%	15.8%
Project 4	21.1%	31.6%	47.4%
Project 5	73.7%	0.0%	26.3%
Project 6	57.9%	15.8%	26.3%
Project 7	47.4%	21.1%	31.6%
Project 8	42.1%	47.4%	10.5%
Project 10	36.8%	5.3%	57.9%
Project 12	42.1%	57.9%	0.0%
Project 13	31.6%	42.1%	26.3%
Project 14	42.1%	15.8%	42.1%
Project 15	52.6%	31.6%	15.8%

CHAPTER 5. CONCLUSION AND FUTURE WORK

Complete and unambiguous security requirements defined prior to deploying a software development project will result in the reduction of defects, early discovery of errors, generation of guidelines for the future creation of requirements, and lastly, enablement of software providers and stakeholders to identify and request the necessary security features. These demonstrate the need for a formal technique to analyze and evaluate security requirements. This research seeks to identify the degree of incompleteness and ambiguity of security requirements, assisting with security compliance before moving to the software development phase.

Using NLP and ML-based tools, i.e., textual entailment and neural network modeling, a given security requirements document can be evaluated against the security standards in order to determine the level of completeness. In order to determine the best entailment configuration out of the 10 options, the ability to correctly predict the completeness (based on the average F-scores across all 13 analyzed projects) was evaluated for each entailment configuration. The results demonstrate that entailment configuration 9 has the highest average F-score, 0.70, and thus, it is the best predictor of completeness.

Proposed future work for this research includes adding additional operators such as contradiction, restricting the standard statements to only those that apply to the given software project, analyzing additional non-functional software requirements such as usability and maintainability, and finally, modifying the missing and ambiguous requirements by receiving appropriate feedback to complete the requirements specification and transfer it to the next phase of development.

REFERENCES

REFERENCES

Ambler, S. (n.d.). *Examining the Agile Cost of Change Curve*. Retrieved 2016, from Ambysoft: http://www.ambysoft.com/essays/whyAgileWorksFeedback.html.

Bucchiarone, A., Fantechi, A., Gnesi, S. L., & Trentanni, G. (2008). QuARS Express - An automatic analyzer of natural language requirements. *Proceedings of 23rd IEEE/ACM International Conference on Automated Software Engineering*, (pp. 473-474).

Casamayor, A., Godoy, D., & Campo, M. (2010). Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology 52*, (pp. 436-445).

Chollet, F. (2015). *Keras*. Retrieved 2016, from https://keras.io/.

Cleland-Huang, J., Settimi, R., Xuchang, Z., & Solc, P. (2006). The Detection and Classification of Non-Functional Requirements with Application to Early Aspects. *14th IEEE International Requirements Engineering Conference (RE'06).* Minneapolis/St. Paul, MN.

Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2007). Automated classification of non-functional requirements. *Requir. Eng. 12* (pp. 103-120). DOI=http://dx.doi.org/10.1007/s00766-007-0045-1.

Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (August, 2007). Automated Detection and Classification of Quality Requirements. *Requirements Engineering Journal, Springer-Verlag*, 36-45.

Doerr, J., Kerkow, D., Koenig, T., Olsson, T., & Suzuki, T. (2005). Non-functional requirements in industry - Three case studies adopting an experience-based NFR method. *13th IEEE Int. Conf. on Requirements Engineering.*

Fisher, J. (2007, September 10). *Owasp Application Security Requirements.* Retrieved 2015, from https://www.owasp.org/index.php/File:OWASP_Application_Security_Requirements_-_Identification_and_Authorisation_v0.1_(DRAFT).doc.

Giampiccolo, D., Magnini, B., & Szpektor, I. (2006). *The Second PASCAL Recognising Textual Entailment Challenge.*

Gnesi, S. F. (2005). *An automatic tool for the analysis of natural language requirements.* Leicester: CRL Publishing.

Hooper, D., Couglan, J., & Mullen, M. (2008). Structural equation modelling: guidelines for determining model fit. *Electronic Journal of Business Research Methods, 6(1)*, 53-60.

ISO. (2009). *Evaluation, ISO/IEC 15408: Information technology - Security techniques*. Retrieved 2015, from http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm.

ISO. (2015, January). *ISO/IEC 27001 - Information security management*. Retrieved from International Organization for Standardization: http://www.iso.org/iso/home/standards/management-standards/iso27001.htm.

Javed, T., Maqsood, M. e., & Durrani, Q. S. (2004, May). A Study to Investigate the Impact of Requirements Instability on Software Defects. *SIGSOFT Softw. Eng. Notes, 29*, 1-7. doi:10.1145/986710.986727.

Kamsties, E., Berry, D. M., & Paech, B. (2001). Detecting ambiguities in requirements documents using inspections. *In Proceedings of the first workshop on inspection in software engineering (WISE'01)*, (pp. 68-80).

Karg, L., & Beckhaus, A. (2008). Analysis of software quality cost modeling's industrial applicability with focus on defect estimation. *IEEE International Conference on Industrial Engineering and Engineering Management*, (pp. 287-291). Singapore.

Kassab, M., Daneva, M., & and Ormandjieva, O. (2007). *Early quantitative assessment of non-functional requirements.* University of Twente Report.

Kouylekov, M., & Magnini, B. (2005). Recognizing Textual Entailment with Tree Edit Distance Algorithms. Trento, Italy.

MacDonell, S., Min, K., & Connor, A. (2005). Autonomous Requirements Specification Processing Using Natural Language Processing. *Proceedings of the 14th International Conference on Adaptive Systems and Software Engineering (IASSE05)*, (pp. 266-270).

Magnini, B., Zanoli, R., Dagan, I., Eichler, K., Neumann, G., Noh, T., . . . Levy, O. (2014). The Excitement Open Platform for Textual Inferences. *In ACL (System Demonstrations)*, (pp. 43-48).

Malhotra, R., Chug, A., Hayrapetian, A., & Raje, R. (2016). Analyzing and evaluating security features in software requirements. *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)* (pp. 26-30). Noida: doi: 10.1109/ICICCS.2016.7542334.

Mehdad, Y., & Magnini, B. (2009). Optimizing Textual Entailment Recognition Using Particle Swarm Optimization. *Proceedings of the 2009 Workshop on Applied Textual Inference* (pp. 36–43). Suntec, Singapore: ACL-IJCNLP.

Mohamed Farid, W. (2011). The NORMAP Methodology: Non-functional Requirements Modeling for Agile Processes. Doctoral dissertation. Retrieved 2015, from http://nsuworks.nova.edu/gscis_etd/147.

Nigam, K., Lafferty, J., & McCallum, A. (1999). Using Maximum Entropy for Text Classification. *IJCAI-99 Workshop on Machine Learning for Information Filtering, vol 1*, (pp. 61–67). Pittsburgh, PA.

Nivre, J. (2008). Algorithms for Deterministic Incremental Dependency Parsing. *Computational Linguistics* (pp. 513-553). vol. 34, no. 4.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., . . . Marsi, E. (2007). MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering, 13(2)* (pp. 95–135). doi: 10.1017/S13513249.

OWASP, ". A. (2015, January). *Category:OWASP Application Security Verification Standard Project*. Retrieved from OWASP: https://www.owasp.org/index.php/Main_Page.

PCI Security Standards Council LLC. (2010, Oct). *Requirements and Security Assessment Procedures.* Retrieved January 2015, from Payment Card Industry (PCI) Data Security Standard: https://www.pcisecuritystandards.org/document_library.

Raje, R., & Malhotra, R. (2015, May 27-28). *Abstract Catalog.* Retrieved June 2015, from S2ERC, An NSF Industry/University Cooperative Research Center: http://www.serc.net.

Rojas, A., & Sliesarieva, G. (2010). Automated detection of language issues affecting accuracy, ambiguity and verifiability in software requirements written in natural language. *Proceedings of the NAACL HLT Young Investigators Workshop on Computational Approach*, (pp. 100-108).

Russell, S., & Norvig, P. (1995). Artificial Intelligence A Modern Approach. Englewood Cliffs, New Jersey : Alan Apt.

Schmid, H. (1994). Probablistic Part-of-Speech Tagging Using Decision Trees. *in Proceedings of International Conference on New Methods in Language Processing.* Manchester, UK.

Standards. (2015, January). Retrieved from ISO: http://www.iso.org/iso/home.htm.

Takahashi, T., Kannisto, J., Harju, J., Kanaoka, A., Takano, Y., & Matsuo, S. (2014). Expressing Security Requirements: Usability of Taxonomy-Based Requirement Identification Scheme, Services (SERVICES). *IEEE World Congress on*, (pp. 121-128).

The Apache Software Foundation. (2016). *Apache OpenNLP*. Retrieved April 2016, from OpenNLP: https://opennlp.apache.org.

Wilson, W., Rosenberg, L., & Hyatt, L. (1997). Automated analysis of requirement specifications. *Proceedings of the 19th ACM international conference on Software engineering*, (pp. 161-171).

Zanoli, R. (2015, May). *EditDistance*. Retrieved April 2016, from Wiki for EOP-1.2.3 release: https://github.com/hltfbk/EOP-1.2.3/wiki/EditDistance.

PUBLICATIONS

PUBLICATIONS

R. Malhotra, A. Chug, A. Hayrapetian and R. Raje, "Analyzing and evaluating security features in software requirements," *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, Noida, 2016, pp. 26-30.doi: 10.1109/ICICCS.2016.7542334.

A. Mangaonkar, A. Hayrapetian and R. Raje, "Collaborative detection of cyberbullying behavior in Twitter data," *2015 IEEE International Conference on Electro/Information Technology (EIT)*, Dekalb, IL, 2015, pp. 611-616.
doi: 10.1109/EIT.2015.7293405.