AI Requirements Identification and Classification for Systems Engineering

Final Report

Omid Shirazi, Megan Taylor, Michael Tzimourakas, Alex Zepka

OR/SYST 699

Sponsor: MITRE

May 7, 2021

Table of Contents

[1. Executive Summary 1](#_Toc70944877)

[2. Project Definition 1](#_Toc70944878)

[3. Systems Engineering 1](#_Toc70944879)

[4. Work Breakdown Structure 1](#_Toc70944880)

[5. Context Analysis 3](#_Toc70944881)

[6. Stakeholder Analysis 6](#_Toc70944882)

[7. Problem and Need Statements 6](#_Toc70944883)

[8. Concept of Operations 7](#_Toc70944884)

[9. Graphical User Interface 7](#_Toc70944885)

[10. Requirements 8](#_Toc70944886)

[11. Verification & Validation Testing 13](#_Toc70944887)

[12. Methodology for Developing AI Prototype 16](#_Toc70944888)

[13. Model Design 18](#_Toc70944889)

[14. Implementation 22](#_Toc70944890)

[15. Analysis of Strategy Alternatives 29](#_Toc70944891)

[16. Prototype 30](#_Toc70944892)

[17. Conclusion 32](#_Toc70944893)

[18. Future Work 32](#_Toc70944894)

[19. Challenges 32](#_Toc70944895)

[20. References 33](#_Toc70944896)

Table of Figures

[Figure 1: As-Is Process 3](#_Toc70944909)

[Figure 2: Vee Model 4](#_Toc70944910)

[Figure 3: Supervised Neural Network Graph 6](#_Toc70944911)

[Figure 4: Concept of Operations 7](#_Toc70944912)

[Figure 5: GUI Concept 8](#_Toc70944913)

[Figure 6: Functional Architecture 9](#_Toc70944914)

[Figure 7: Sentence Slicing system Activity Diagram 10](#_Toc70944915)

[Figure 8: Training system Activity Diagram 11](#_Toc70944916)

[Figure 9: Data Pre-processor system Activity Diagram 11](#_Toc70944917)

[Figure 10: Testing system Activity Diagram 12](#_Toc70944918)

[Figure 11: AI Methodology Activity Diagram 16](#_Toc70944919)

[Figure 12: BERT Layer diagram 19](#_Toc70944920)

[Figure 13: FNN diagram 21](#_Toc70944921)

[Figure 14: CNN diagram 22](#_Toc70944922)

[Figure 15: BERT model diagram 22](#_Toc70944923)

[Figure 16: Sensitivity vs Specificity graph 27](#_Toc70944924)

[Figure 17: Sentences given to reviewer vs true sentences identified graph 30](#_Toc70944925)

Table of Tables

[Table 1: Work Breakdown Structure 3](#_Toc70944926)

[Table 2: Stakeholder Map 6](#_Toc70944927)

[Table 3: Problem and Need Statements 7](#_Toc70944928)

[Table 4: Requirements Verification Matrix 16](#_Toc70944929)

[Table 5: Dataset Library 18](#_Toc70944930)

[Table 6: Threshold dataset and layer examples 20](#_Toc70944931)

[Table 7: Confusion Matrix example 24](#_Toc70944932)

[Table 8: Confusion Matrix results for FNN model 24](#_Toc70944933)

[Table 9: Confusion Matrix results for CNN model 25](#_Toc70944934)

[Table 10: Confusion Matrix results for BERT layer 25](#_Toc70944935)

[Table 11: Confusion Matrix results for 1 Out of 3 model 25](#_Toc70944936)

[Table 12: Confusion Matrix results for 2 Out of 3 model 25](#_Toc70944937)

[Table 13: Confusion Matrix results for 3 Out of 3 model 25](#_Toc70944938)

[Table 14: Confusion Matrix results for CNN & BERT Agree model 25](#_Toc70944939)

[Table 15: Confusion Matrix results for FNN & BERT Agree model 25](#_Toc70944940)

[Table 16: Accuracy results for various models 26](#_Toc70944941)

[Table 17: Sensitivity and Specificity Results 26](#_Toc70944942)

[Table 18: F1 Score results of various models 27](#_Toc70944943)

[Table 19: MCC results of various models 28](#_Toc70944944)

[Table 20: Bayes probability results for various models 28](#_Toc70944945)

[Table 21: Overall results of various models 29](#_Toc70944946)

[Table 22: Sample Algorithm Output 31](#_Toc70944947)

[Table 23: Sample Algorithm Output with Highlights 32](#_Toc70944948)

# Executive Summary

Systems Engineering: Requirements Identification​

* Systems Engineering is a complex, time-consuming, and tedious process that relies on a lot of attention to detail. Often, a reviewer goes through 100+ documents to elicit requirements​.
* As-Is Process: Review reads through requirements document to copy and paste the requirements into a spreadsheet.​

Problem and Need​

* Lengthy process of eliciting requirements from the requirements document.​
* Current Requirement Analysis Tools do not cover this problem.​
* Need to determine if AI can be leveraged in a way to increase the quality and accuracy of the engineering process and can it improve outcomes.

To-Be Process​

* Feed requirements document into system, split into sentences, and let AI predict whether the sentence is a requirement. Output spreadsheet of requirements.​

Design​

* Build 3 different Neural Networks to compare and combine results to get the requirement predictions​.

Results​

* Reduce workload by 50% and elicit 90% of true requirements​.
* Reduce workload by 80% and elicit 50% of true requirements.

# Project Definition

MITRE has asked the George Mason University Systems Engineering and Operations Research office to identify an inefficiency in the practice of the systems engineering development process that artificial intelligence (AI) can assist and improve. Discussions with the sponsor and a literature review reveal that the requirements definition process is an ideal opportunity for AI to integrate into and help improve the efficiency of the review process.

# Systems Engineering

The systems engineering format that was chosen for this system development was the Vee model. The reasons for this are: the sponsor put a high value on the system fulfilling the objectives created; the team had frequent, direct communication with the sponsor in order to flesh out the validation criteria early in the program; the team could perform trades to bound the system and its requirements. A two-loop development cycle was also established in order to burn down risk in the first prototype and receive feedback that would be used to understand the parameters for the final design.

# Work Breakdown Structure

The Work Breakdown Structure (WBS) was created for this project to quantify and scope the work needed to accomplish the objectives of the project. This WBS and accompanying schedule was presented to the customer representatives to ensure communication of what could realistically be accomplished in the span of a semester was understood by all stakeholders. The WBS is shown below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **WBS** | **Description** | **Duration** | **Start Date** | **Completion Date** |
| 1 | Class Deliverables | 85 days | 2/11/2021 | 5/6/2021 |
| 1.1 | Draft Proposal | 1 day | 2/11/2021 | 2/11/2021 |
| 1.2 | Final Proposal | 1 day | 3/4/2021 | 3/4/2021 |
| 1.3 | Report Outline | 1 day | 3/25/2021 | 3/25/2021 |
| 1.4 | Preliminary Report | 1 day | 4/15/2021 | 4/15/2021 |
| 1.5 | SAGE Manuscript | 1 day | 4/19/2021 | 4/19/2021 |
| 1.6 | Final Report | 1 day | 4/29/2021 | 4/29/2021 |
| 1.7 | Final Presentation | 1 day | 5/6/2021 | 5/6/2021 |
| 2 | Context Analysis | 30 days | 2/1/2021 | 3/2/2021 |
| 2.1 | Research | 30 days | 2/1/2021 | 3/2/2021 |
| 2.1.1 | Gap Analysis | 4 days | 2/1/2021 | 2/4/2021 |
| 2.1.2 | AI integration | 30 days | 2/1/2021 | 3/2/2021 |
| 2.2 | Stakeholder Interaction | 11 days | 2/1/2021 | 2/11/2021 |
| 2.2.1 | System Scope Discussion | 8 days | 2/1/2021 | 2/8/2021 |
| 2.2.2 | Document Procurement | 11 days | 2/1/2021 | 2/11/2021 |
| 2.2.3 | Meeting Planning | 1 day | 2/1/2021 | 2/1/2021 |
| 3 | Initial Prototype | 37 days | 2/12/2021 | 3/20/2021 |
| 3.1 | Design | 23 days | 2/12/2021 | 3/6/2021 |
| 3.1.1 | Requirement Definition | 5 days | 2/16/2021 | 2/20/2021 |
| 3.1.2 | Pre-process Datasets | 14 days | 2/12/2021 | 2/25/2021 |
| 3.1.3 | Build Prototype | 7 days | 2/21/2021 | 2/27/2021 |
| 3.1.4 | Train Prototype | 7 days | 2/28/2021 | 3/6/2021 |
| 3.2 | Test | 18 days | 2/26/2021 | 3/15/2021 |
| 3.2.1 | Create Verification Test Plan | 11 days | 3/2/2021 | 3/12/2021 |
| 3.2.2 | Create Test Dataset | 9 days | 2/26/2021 | 3/6/2021 |
| 3.2.3 | Conduct Test | 3 days | 3/13/2021 | 3/15/2021 |
| 3.3 | Retrospection | 5 days | 3/16/2021 | 3/20/2021 |
| 3.3.1 | Analyze Test Results | 2 days | 3/16/2021 | 3/17/2021 |
| 3.3.2 | Stakeholder Feedback | 1 day | 3/18/2021 | 3/18/2021 |
| 3.3.3 | Map Changes to Final Prototype | 2 days | 3/19/2021 | 3/20/2021 |
| 4 | Final Prototype | 47 days | 3/21/2021 | 5/6/2021 |
| 4.1 | Design | 16 days | 3/21/2021 | 4/5/2021 |
| 4.1.1 | Requirement Definition | 2 days | 3/21/2021 | 3/22/2021 |
| 4.1.2 | Pre-process Datasets | 7 days | 3/21/2021 | 3/27/2021 |
| 4.1.3 | Build Prototype | 7 days | 3/23/2021 | 3/29/2021 |
| 4.1.4 | Train Prototype | 7 days | 3/30/2021 | 4/5/2021 |
| 4.2 | Test | 3 days | 4/6/2021 | 4/8/2021 |
| 4.2.1 | Create Verification and Validation Test Plan | 3 days | 4/6/2021 | 4/8/2021 |
| 4.2.2 | Create Test Dataset | 3 days | 4/6/2021 | 4/8/2021 |
| 4.2.3 | Conduct Sponsor Test | 3 days | 4/6/2021 | 4/8/2021 |
| 4.3 | Retrospection | 28 days | 4/9/2021 | 5/6/2021 |
| 4.3.1 | Analyze Test Results | 28 days | 4/9/2021 | 5/6/2021 |
| 4.3.2 | Validate Test Results | 28 days | 4/9/2021 | 5/6/2021 |
| 5 | Program Closeout | 1 day | 5/7/2021 | 5/7/2021 |
| 5.1 | Software Handover | 1 day | 5/7/2021 | 5/7/2021 |
| 5.2 | Document Handover | 1 day | 5/7/2021 | 5/7/2021 |

Table 1: Work Breakdown Structure

# Context Analysis

## Background

The proposal review process is a necessary and time-consuming process from the perspective of the solicitors that must sift through often dozens of responses, each one containing hundreds of pages of historical performance, future forecasts, costs, and proposed specifications. Government solicitation review teams are composed of the core government team that wrote the offer, followed by a coalition of domain experts from other agencies/departments and FFRDCs (Federally Funded Research and Development Centers). Many review teams are undermanned, requiring each reviewer to analyze many proposals in a short amount of time. Reviewers are also held to the FAR (Federal Acquisition Regulation) system which guides how the proposals are to be evaluated and can lengthen the review process in order to ensure a fair and unbiased review.

## As-Is Process

In the AS-IS process of requirements definition, the reviewer would take the lengthy document, often over 250 pages, and read or search for key words. They take that relevant text and cut and paste it into an Excel spreadsheet. Two gaps are identified here. First, the reviewer may not be a domain expert to the system at hand so they could miss relevant information or take information that is not relevant. The first causing problems with communication and getting a complete understanding of the system, and the latter wasting time with irrelevant text. Second, gathering the requirements from a document by cutting and pasting the text is very inefficient and time consuming. Figure 1 illustrates this process in an activity diagram.

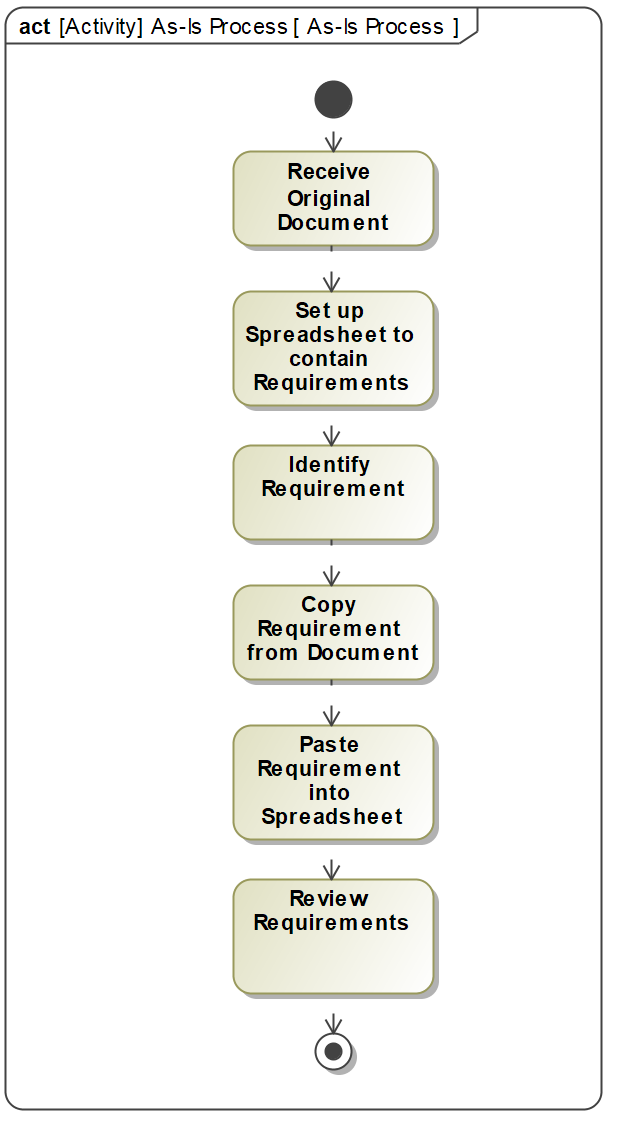


Figure 1: As-Is Process

## Systems Engineering Applications

### Overview

This section details the literature analysis performed to: 1) identify historical examples of AI/ML integration in the systems engineering process, 2) identify opportunity areas to narrow down this paper’s focus of study, and 3) to identify future opportunities for AI/ML integration in the systems engineering process. The system engineering architectural model used for the analysis is the Vee model, as it was made in agreement with the customer sponsor, and is a popular systems engineering architecture framework among the customer sponsor. The Vee model sections are split into: Concept Development; Requirements Engineering; System Architecture; System Design & Development; System Integration; Test & Evaluation; and Transition, Operation, & Maintenance.

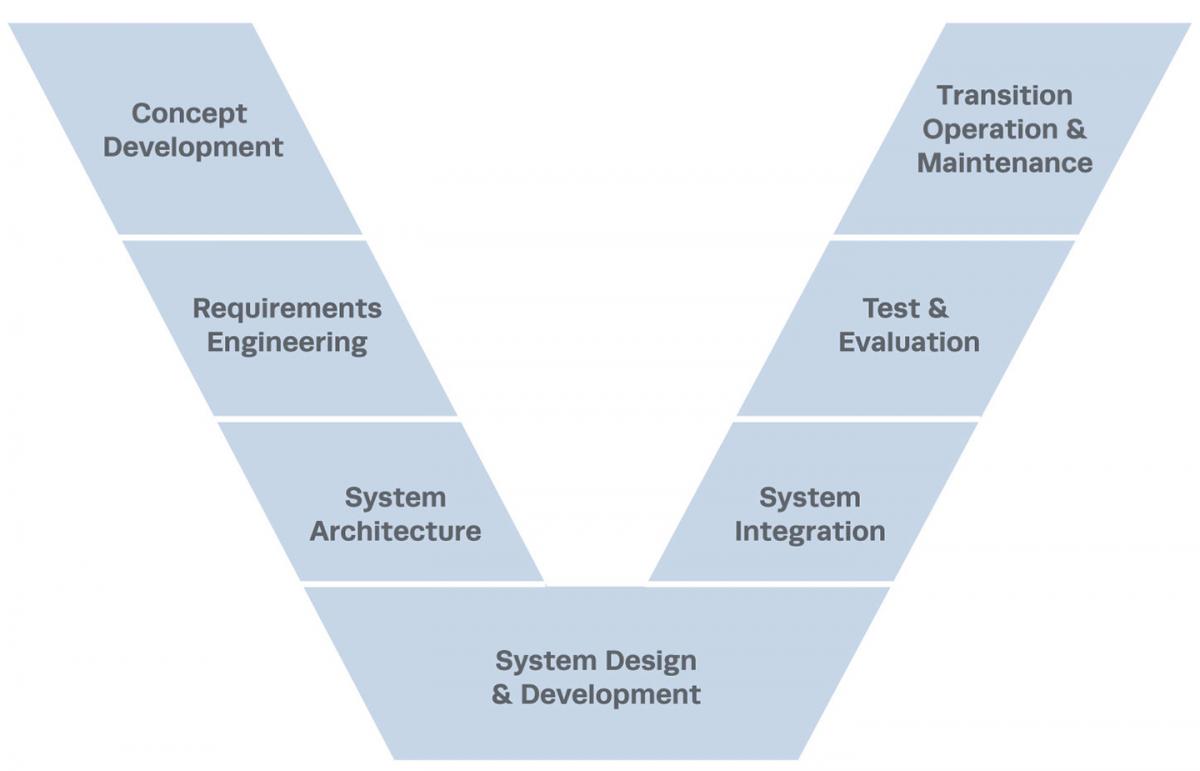


Figure 2: Vee Model [1]

### Concept Development

No historical examples of AI/ML integration were found for the Concept Development phase. Future opportunities in this phase for other studies not included in this paper include AI/ML for: customer objectives and needs identification, analyzing patterns in stakeholder lists of objectives (do multiple stakeholders share similar patterns of what they envision the system to be capable of?), and analyzing past program performance of similar efforts to strategize and optimize trade study subjects.

### Requirements Engineering

There are historical implementations of AI in the requirements engineering process. In “Implementing Augmented Intelligence in Systems Engineering” (2018), AI was used to enhance the user’s ability to trace requirements changes to changes in the physical parameters of the SWAP-C (size, weight, power, and cooling) characteristics of systems [2]. In “Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents,” (2008) an AI tool was created to be able to parse a requirements document using semantic analysis [3]. In “QuARS: A Tool for Analyzing Requirements,” another AI tool was created to be able to detect linguistic inaccuracies in requirements that may lead to vague interpretation of a requirement [4]. KAOS is another popular tool for requirements engineering used to calculate requirements from goal diagrams [5]. These AI/ML solutions to requirements engineering exist to the right of the process, where requirements have already been identified, or heavy work in model-based software has been performed. A literature review did not identify any historical solution of AI/ML used in the requirement identification process.

### System Architecture

No historical examples of AI/ML integration were found for the System Architecture phase. Future opportunities in this phase for other studies not included in this paper include AI/ML for: analysis of historical similar system architectures for areas of highest risk, and comparison of system architecture components to requirements to identify gaps in architecture with too few requirements.

### System Design and Development

Historical examples of AI/ML in the System Design and Development phase are found in “Implementing Augmented Intelligence in Systems Engineering” (2018). Along with capabilities in the requirements engineering phase, the AI would also apply fault isolation strategically to system areas that were prone to faults. A future opportunity in this phase for other studies not included in this paper include AI/ML for: analysis of similar historical systems and their development schedule for most likely areas of schedule/cost/performance slippages.

### System Integration

No historical examples of AI/ML integration were found for the System Integration phase. A future opportunity in this phase for other studies not included in this paper include AI/ML for analysis of system component buildup for high risk areas (hard to access areas for system maintenance during the Test and Evaluation phase, areas with little margin for modification).

### Test and Evaluation

The historical example in “Implementing Augmented Intelligence in Systems Engineering” (2018) applies to this section also, due to its fault isolation analysis capabilities that are utilized during the Test and Evaluation phase. Future opportunities in this phase for other studies not included in this paper include AI/ML for an analysis of similar systems to identify high risk areas of slippage with testing schedules, and a mapping of requirements changes to how they could impact the testing architecture (keeping system design changes tightly mapped to impacts to its Test and Evaluation phase is critical to reduce cost/schedule/performance impacts here).

## Machine Learning

Machine Learning is the subset of Artificial Intelligence that is implemented in this project. It is used to elicit trends or rules out of a dataset to have more accurate decision-support models for predicting specified future trends. It can be done in a supervised or unsupervised manner.

## Machine Learning for Requirements Identification

There are many different machine learning algorithms in the range between sentiment and semantic that can be uses to predict a sentence either as requirement or not. These includes Logistic Regression, Naïve Bayes, Support Vector Machines and Neural Networks. These are shown in the figure below and their relative ordering between sentiment and semantic relative to each other.

Semantic models consider the position of the word and order of them in the sentences. Therefore, they are more accurate than sentiment models which only consider the word itself. In this project three supervised neural network have been used to train models to identify sentences as system’s requirements. These include a Fully Connected Neural Network, a Convolutional Neural Network, and a BERT layer in a Neural Network.

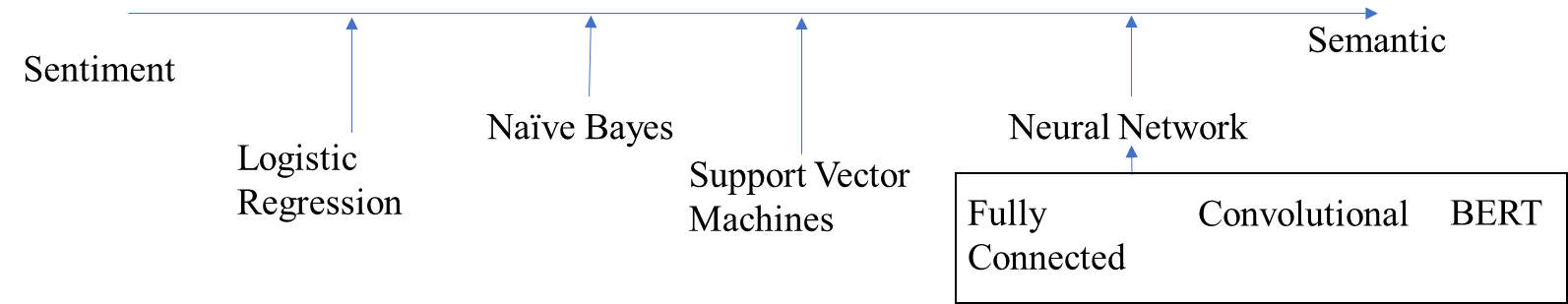


Figure 3: Supervised Neural Network Graph

# Stakeholder Analysis

Three primary stakeholders identified include the review of the original documents, the vendor who submits the original document that needs to be reviewed, and the issuer who requests the original document. The objectives and tensions of the stakeholders are summarized in the following table.

|  |  |  |
| --- | --- | --- |
| Stakeholder | Objectives | Tensions |
| Reviewer | Elicit requirements of the system from the original document | The reviewer wants a precise document but it takes more work load for the vendor to develop. |
| Vendor | Submit original document from which the requirements need to be elicited | The Vendor likes to know the result of review faster, while the reviewer needs time to be accurate enough. |
| Issuer | Request for original document | The Issuer wants to get the best system with the lowest cost, while the vendor prefers a system with minimal requirements to make the most possible profit |

Table 2: Stakeholder Map

# Problem and Need Statements

The problems and needs identified for this project are summarized in the following table.

|  |  |
| --- | --- |
| Problem | Need |
| In Requirements Process:    * 1000’s Requirements * multiple traceability properties to Tests, Demonstrations, and Reports | A way to manage the complexity and volume of requirements and their associated documents. |
| Large changing dataset means it is very costly to dedicate an engineer/s to keep track of the impacts of program modifications to requirements | Implement AI to better manage the requirement process. |
| Reviewer may not be a domain expert to the system at hand causing inefficiency. |

Table 3: Problem and Need Statements

# Concept of Operations

The system out of the box will be capable of reading a document, slicing the sentences, and predicting whether that sentence is a requirement or not. The AI system is already trained with requirements documents and will utilize this training for its prediction. Therefore, a user may ingest any document and the sentences will be classified.

In order for the user to tailor the algorithm for their respective use case, the AI must be retrained with the data that the user wants. This will allow the algorithm to be more accurate for a specific set of documents the user will want the AI to classify. This iterative maintenance of the system will allow the user to create a more accurate system for their specific use case.

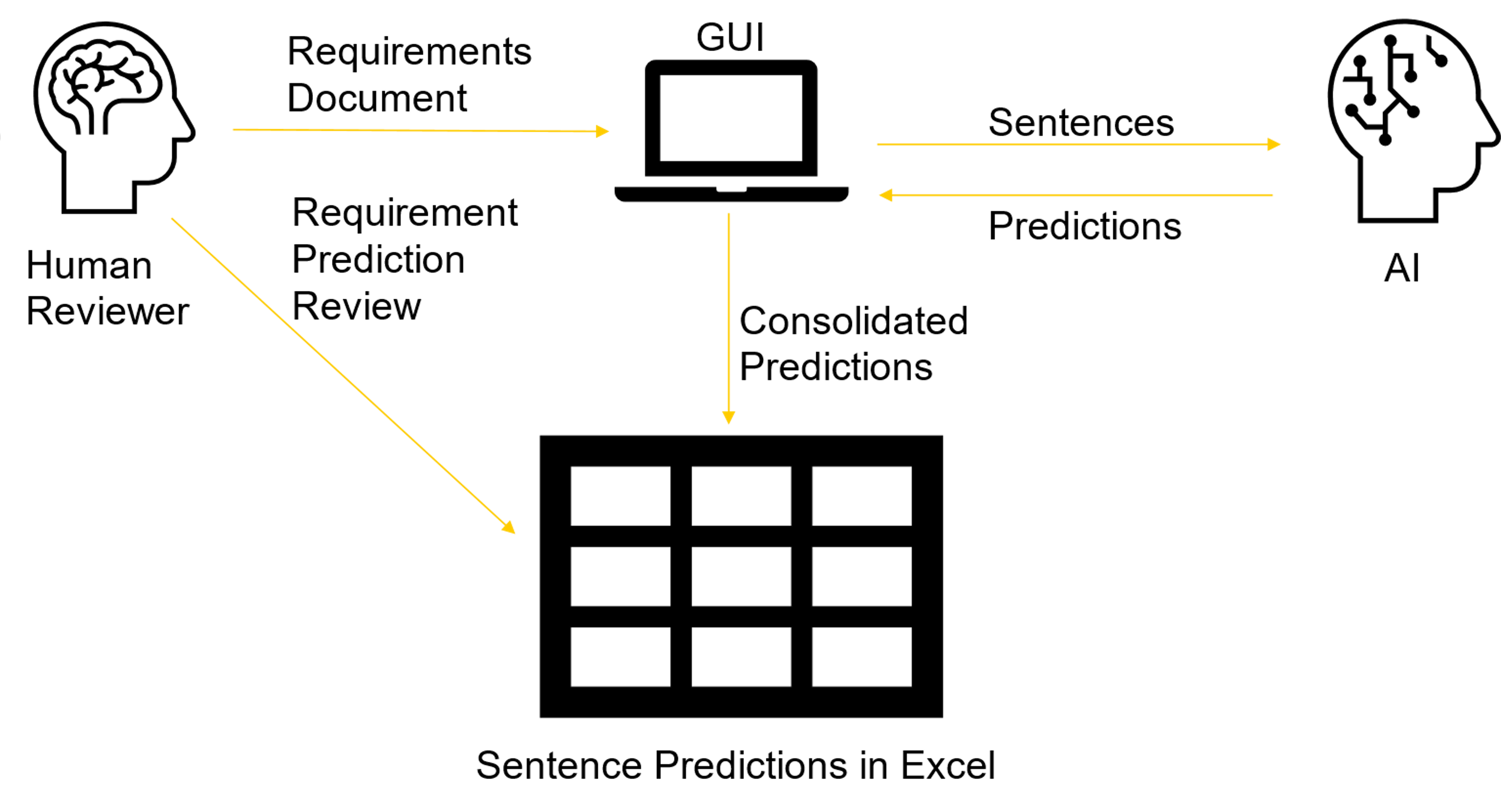


Figure 4: Concept of Operations

# Graphical User Interface

In order for the user to process requirements for a given document, a simple Graphical User Interface (GUI) was created to interface the AI with the user.

Graphical user interface, text, application

Description automatically generated

Figure 5: GUI Concept

The GUI contains a number of functions to allow the user to process the document. First, the user will utilize the file selection function of the GUI and select the respective file they would like processed. This file will be ingested into the AI algorithm to be sliced into sentences, read, and predicted upon. The algorithm will output a detailed report of its analysis. The GUI will be populated with data from the detailed report as initial data such as number of sentences, number of requirements found for each model. The user may open the detailed report using the GUI as well. This user interface acts as a system the user my interact with the AI algorithm without overwhelming the user with the details. Further discussion of the GUI and prototype will be discussed later in the paper.

# Requirements

## Functional Architecture (To-Be Process)

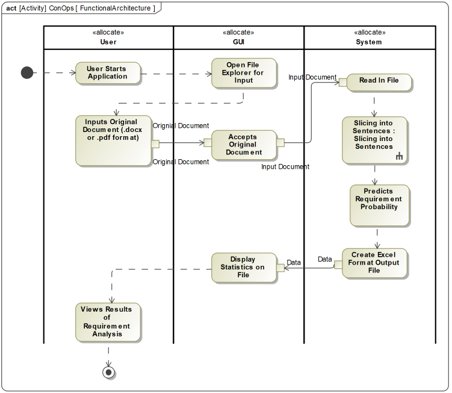


Figure 6: Functional Architecture

The use of the system starts when the user starts the application. A GUI opens with a file explorer for the user to input the original requirement document. Then, the user locates the document, and the GUI accepts it. After that, the file is given to the system to read in the file. The system slices the file, predicts the requirement probability, and creates an Excel output file with the results from each of the models. The data is sent to the GUI to display and the user can then view the results of the requirement identification analysis.

## Mission Requirements

### The system shall slice an original requirements document into individual sentences.

### The system shall predict the probability of being a requirement for each sentence with an MCC Score of at least 0.4.

### The system shall accept an original requirements document from the user.

#### The system shall accept an original requirements document in a PDF format.

#### The system shall accept an original requirements document in a DOCX format.

### The system shall output an Excel file with the requirement probability of each sentence using various models in the original requirements document.

## Design Requirements

### Slicing Sentences

The system has the functionality of slicing the documents into the individual sentences. To do this, it first reads the paragraphs, counts the periods, and then iterates through those periods. When it gets to each period, it needs to decide if that period is at the end of a sentence or not. It does not consider a period to be at the end of a sentence if it is between an i or e, an e and g, or between two numbers or a letter and a number. Any other period is considered at the end of the sentence. This capability is not perfect and has is a great place for future work to be conducted. The following figure shows this process.

Diagram

Description automatically generated

Figure 7: Sentence Slicing system Activity Diagram

#### The system shall identify and read the paragraphs of an original requirements document.

#### The system shall count the periods within the paragraph.

#### The system shall iterate through each of the periods in the paragraph.

#### The system shall identify when the period is the end of the sentence.

* + - * 1. The system shall identify the end of a sentence if the period is between a “)” and a letter.
        2. The system shall identify the end of a sentence if the period is between two letters.
        3. The system shall identify the end of a sentence if the period is between a letter and a space.

#### The system shall identify when the period is not at the end of the sentence.

* + - * 1. The system shall identify that a period is not the end of the sentence if the period is between an “e” and a “g”.
        2. The system shall identify that a period is not the end of the sentence if the period is between two numbers.
        3. The system shall identify that a period is not the end of the sentence if the period is between a letter and a number.

#### The system shall compile and make a list of all the sentences that are identified.

### Training

This project also produced the capability of training the neural networks that were developed. It starts with reading in the labeled training data, then preprocessing it. Preprocessing consists of making the sentences into a list, tokenizing the sentences, making a vocabulary, and then embedding each sentence. Then, the models are defined, and the models can be trained. Within the model definition, the training data is split into training data and validation data. Within each epoch of training, the data is shuffled so that it uses a different validation set each time. All of this is summarized in the following diagrams.

Diagram

Description automatically generated

Figure 8: Training system Activity Diagram

Diagram

Description automatically generated

Figure 9: Data Pre-processor system Activity Diagram

#### The system shall read in a labeled data set.

#### The system shall preprocess the data.

* + - * 1. The system shall make a list of sentences from the input list of data.
        2. The system shall tokenize each sentence.
        3. The system shall make a vocabulary.
        4. The system shall embed each sentence.

#### The system shall define the models.

* + - * 1. The models shall split the data into training and validation data.
        2. The models shall shuffle the training data in each epoch.
        3. The models shall use different validation data in each epoch.
        4. The system shall implement a sentiment model.
        5. The system shall implement a Semantic model.

The system shall contain a BERT layer in at least on model.

#### The system shall train the model using the training data.

### Testing

After each of the models are trained, they are tested. This is done by inputting labeled data that was not trained upon, then separating the sentences from the labels. After this, each sentence is predicted for whether it is a requirement and this is compared to the label given to the requirement. The comparisons allow for the accuracy, F1 score, and MCC to be calculated. This is all in the following diagram.

Diagram

Description automatically generated

Figure 10: Testing system Activity Diagram

#### The system shall accept the labeled test data.

#### The system shall separate the labels from the sentences.

#### The system shall predict the probability that they are a requirement.

#### The system shall compare the probabilities of being a requirement with the original labels.

#### The system shall calculate the test error.

* + - * 1. The system shall calculate the model accuracy,
        2. The system shall calculate the model F1 score.
        3. The system shall calculate the model MCC.

### Output

#### The system shall output sliced sentences from the original requirements document.

#### The system shall output the page number of the sentence from the original requirements document.

#### The system shall output the requirement identification for each sentence.

* + - * 1. The system shall output the requirement identification for each sentence using the FNN model.
        2. The system shall output the requirement identification for each sentence using the CNN model.
        3. The system shall output the requirement identification for each sentence using the semantic model that implements a BERT layer.
        4. The system shall output the requirement identification for each sentence with a prediction that identifies the sentence as a requirement if at least one base model identifies the sentence as a requirement.
        5. The system shall output the requirement identification for each sentence with a prediction that identifies the sentence as a requirement if at least two base models identify the sentence as a requirement.
        6. The system shall output the requirement identification for each sentence with a prediction that identifies the sentence as a requirement if all three base models identify the sentence as a requirement.
        7. The system shall output the requirement identification for each sentence with a prediction that identifies the sentence as a requirement if both the semantic model and the CNN model identify the sentence as a requirement.
        8. The system shall output the requirement identification for each sentence with a prediction that identifies the sentence as a requirement if both the semantic model and the FNN model identify the sentence as a requirement.

# Verification & Validation Testing

The verification test plan matrix is shown below. The matrix maps requirements shown in the Requirements section to methods of verification, which are: analysis, inspection, demonstration, and test. The matrix also maps requirements that are verified via demonstration or test to test events in the WBS mapping. All requirements were verified as of the conclusion of this project. Validation testing was performed as part of WBS 4.2.3 “Conduct Sponsor Test”. Test results were compiled and presented to the customer representatives.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Description** | **Verification Method** | **Verification Artifact** |
| 10.2.1 | The system shall slice an original requirements document into individual sentences. | Test | 3.1.2 |
| 10.2.2 | The system shall predict the probability of being a requirement for each sentence with an MCC Score of at least 0.4. | Test | 4.2.3 |
| 10.2.3 | The system shall accept an original requirements document from the user. | Demonstration | 3.2.3 |
| 10.2.3.1 | The system shall accept an original requirements document in a PDF format. | Demonstration | 3.2.3 |
| 10.2.3.2 | The system shall accept an original requirements document in a DOCX format. | Demonstration | 3.2.3 |
| 10.2.4 | The system shall output a CSV file with the requirement probability of each sentence using various models in the original requirements document. | Test | 4.2.3 |
| 10.3.1.1 | The system shall identify and read the paragraphs of an original requirements document. | Demonstration | 3.1.2 |
| 10.3.1.2 | The system shall count the periods within the paragraph. | Demonstration | 3.1.2 |
| 10.3.1.3 | The system shall iterate through each of the periods in the paragraph. | Demonstration | 3.1.2 |
| 10.3.1.4 | The system shall identify when the period is the end of the sentence. | Demonstration | 3.1.2 |
| 10.3.1.4.1 | The system shall identify the end of a sentence if the period is between a “)” and a letter. | Demonstration | 3.1.2 |
| 10.3.1.4.2 | The system shall identify the end of a sentence if the period is between two letters. | Demonstration | 3.1.2 |
| 10.3.1.4.3 | The system shall identify the end of a sentence if the period is between a letter and a space. | Demonstration | 3.1.2 |
| 10.3.1.5 | The system shall identify when the period is not at the end of the sentence. | Demonstration | 3.1.2 |
| 10.3.1.5.1 | The system shall identify that a period is not the end of the sentence if the period is between an “e” and a “g”. | Demonstration | 3.1.2 |
| 10.3.1.5.2 | The system shall identify that a period is not the end of the sentence if the period is between two numbers. | Demonstration | 3.1.2 |
| 10.3.1.5.3 | The system shall identify that a period is not the end of the sentence if the period is between a letter and a number. | Demonstration | 3.1.2 |
| 10.3.1.6 | The system shall compile and make a list of all the sentences that are identified. | Demonstration | 3.1.2 |
| 10.3.2.1 | The system shall read in a labeled data set. | Demonstration | 3.1.4 |
| 10.3.2.2 | The system shall preprocess the data. | Demonstration | 3.1.4 |
| 10.3.2.2.1 | The system shall make a list of sentences from the input list of data. | Demonstration | 3.1.4 |
| 10.3.2.2.2 | The system shall tokenize each sentence. | Demonstration | 3.1.4 |
| 10.3.2.2.3 | The system shall make a vocabulary. | Inspection | N/A |
| 10.3.2.2.4 | The system shall embed each sentence. | Demonstration | 3.1.4 |
| 10.3.2.3 | The system shall define the models. | Inspection | N/A |
| 10.3.2.3.1 | The models shall split the data into training and validation data. | Demonstration | 3.1.4 |
| 10.3.2.3.2 | The models shall shuffle the training data in each epoch. | Inspection | N/A |
| 10.3.2.3.3 | The models shall use different validation data in each epoch. | Inspection | N/A |
| 10.3.2.3.4 | The system shall implement a sentiment model. | Inspection | N/A |
| 10.3.2.3.5 | The system shall implement a Semantic model. | Inspection | N/A |
| 10.3.2.3.5.1 | The system shall contain a BERT layer in at least on model. | Inspection | N/A |
| 10.3.2.4 | The system shall train the model using the training data. | Inspection | N/A |
| 10.3.3.1 | The system shall accept the labeled test data. | Demonstration | 4.2.2 |
| 10.3.3.2 | The system shall separate the labels from the sentences. | Demonstration | 4.2.2 |
| 10.3.3.3 | The system shall predict the probability that they are a requirement. | Test | 4.2.3 |
| 10.3.3.4 | The system shall compare the probabilities of being a requirement with the original labels. | Test | 4.2.3 |
| 10.3.3.5 | The system shall calculate the test error. | Test | 4.2.3 |
| 10.3.3.5.1 | The system shall calculate the model accuracy, | Test | 4.2.3 |
| 10.3.3.5.2 | The system shall calculate the model F1 score. | Test | 4.2.3 |
| 10.3.3.5.3 | The system shall calculate the model MCC. | Test | 4.2.3 |
| 10.3.4.1 | The system shall output sliced sentences from the original requirements document. | Demonstration | 4.2.3 |
| 10.3.4.2 | The system shall output the page number of the sentence from the original requirements document. | Demonstration | 4.2.3 |
| 10.3.4.3 | The system shall output the requirement identification for each sentence. | Demonstration | 4.2.3 |
| 10.3.4.3.1 | The system shall output the requirement identification for each sentence using the FNN model. | Demonstration | 4.2.3 |
| 10.3.4.3.2 | The system shall output the requirement identification for each sentence using the CNN model. | Demonstration | 4.2.3 |
| 10.3.4.3.3 | The system shall output the requirement identification for each sentence using the semantic model that implements a BERT layer. | Demonstration | 4.2.3 |
| 10.3.4.3.4 | The system shall output the requirement identification for each sentence with a prediction that identifies the sentence as a requirement if at least one base model identifies the sentence as a requirement. | Demonstration | 4.2.3 |
| 10.3.4.3.5 | The system shall output the requirement identification for each sentence with a prediction that identifies the sentence as a requirement if at least two base models identify the sentence as a requirement. | Demonstration | 4.2.3 |
| 10.3.4.3.6 | The system shall output the requirement identification for each sentence with a prediction that identifies the sentence as a requirement if all three base models identify the sentence as a requirement. | Demonstration | 4.2.3 |
| 10.3.4.3.7 | The system shall output the requirement identification for each sentence with a prediction that identifies the sentence as a requirement if both the semantic model and the CNN model identify the sentence as a requirement. | Demonstration | 4.2.3 |
| 10.3.4.3.8 | The system shall output the requirement identification for each sentence with a prediction that identifies the sentence as a requirement if both the semantic model and the FNN model identify the sentence as a requirement. | Demonstration | 4.2.3 |

Table 4: Requirements Verification Matrix

# Methodology for Developing AI Prototype

The steps for the methodology are described in the next sections of the paper. First, the dataset was found and cleaned for preprocessing. The dataset is described further in the paper, but cleaning the data encompasses the process of splitting a requirements document into the individual sentences. The data was preprocessed by labeling whether each sentence in the data was a requirement. Then, different algorithms were identified, developed, trained and evaluated iteratively until the final models were determined.

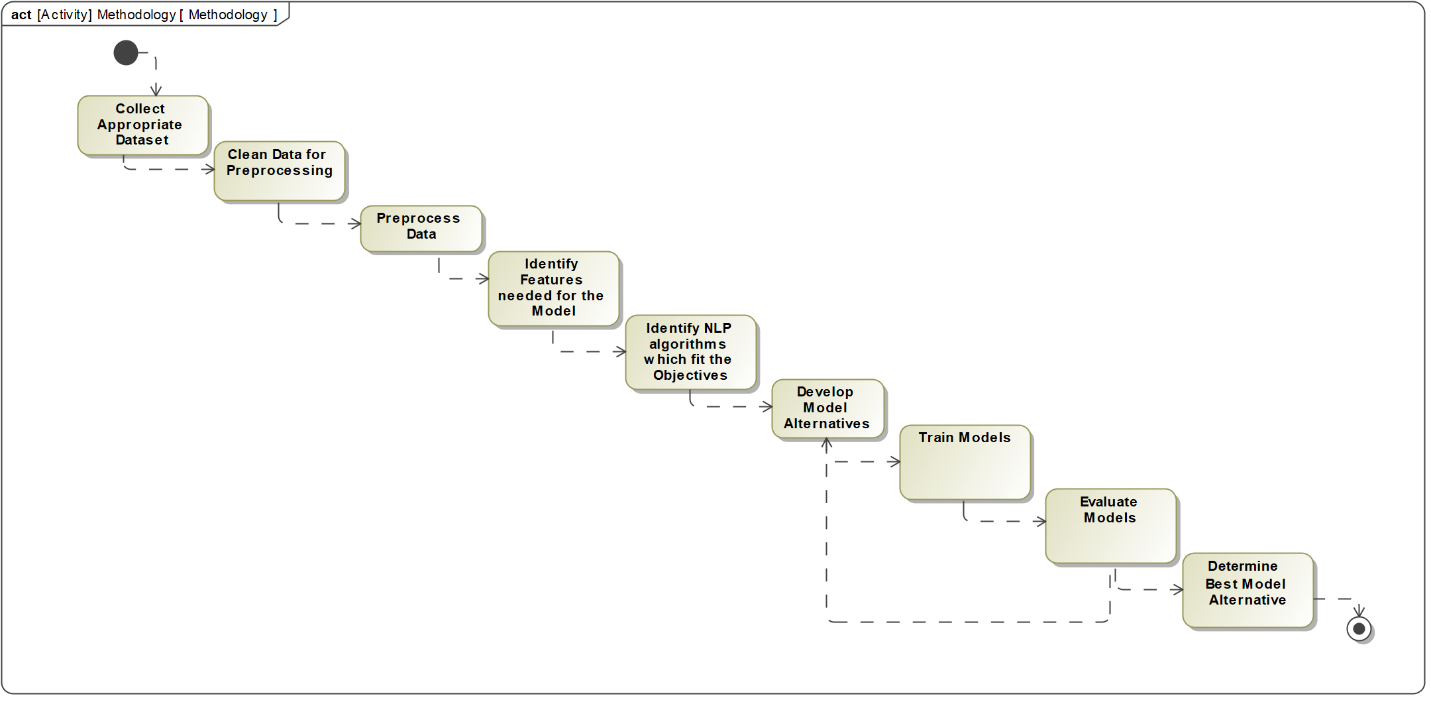


Figure 11: AI Methodology Activity Diagram

## Objectives

### Ability to train the algorithms

### Take in a DOCX or PDF original requirements document.

### Separate the document into the sentences within the document.

### Predict whether a sentence is a requirement or not.

### Return an Excel spreadsheet that contains the predictions for the sentences and the identification of a requirement or not based on a threshold.

## Assumptions

### The labeling of data consisted of those statements that were explicit requirements as well as those sentences that were functions of the system. This is an attempt to capture all aspects of a system that may become a requirement.

### This process does not cover the whole requirements analysis process but is an attempt to reduce the time it takes to search entire documents for requirements.

### The models are not able to differentiate different subjects of requirements, just identify that they are requirements.

## Data Collection

### **Data Source**

Public data of natural language requirements documents were utilized to train the algorithm. This repository contains hundreds of MS Word, PDF, and XML files of requirements documents [6]. Each document contains a system and its respective requirements.

### **Data Preprocessing**

In order for the algorithm to properly learn what sentences are requirements and which are not, requirements documents sentences must be sorted and flagged by a human. Each sentence from the requirements document was extracted and exported into an Excel file one row for each sentence, and one column of sentences. A second column contains a binary flag, 0 if the sentence is not a requirement, 1 if the sentence is a requirement.

This table shows a summary of the documents and the number of labeled sentences with the proportion of labeled requirements and non-requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| Document​ | 1​ | 0​ | Total Sentences |
| 2000 - nasa x38​ | 530​ | 3691​ | 4221​ |
| 2001 - libra​ | 112​ | 312​ | 424​ |
| 2001 - NPAC​ | 1747​ | 6819​ | 8566​ |
| 2004 - ijis​ | 133​ | 204​ | 337​ |
| 2004 - jse​ | 34​ | 396​ | 430​ |
| 2004 - sprat​ | 129​ | 388​ | 517​ |
| 2005 - clarus high​ | 140​ | 314​ | 454​ |
| 2005 - triangle​ | 73​ | 529​ | 602​ |
| 2006 - stewards​ | 88​ | 931​ | 1019​ |
| 2007 - e-store​ | 105​ | 71​ | 176​ |
| 2007 - water use​ | 282​ | 776​ | 1058​ |
| 2008 - peering​ | 24​ | 476​ | 500​ |
| 2008 - viper​ | 155​ | 177​ | 332​ |
| 2008 - virtual\_ed​ | 325​ | 207​ | 535​ |
| 2009 - email​ | 96​ | 309​ | 405​ |
| 2009 - library​ | 66​ | 302​ | 368​ |
| 2009 - video search​ | 24​ | 234​ | 258​ |
| Grand Total​ | 4063​ | 16136​ | 20202​ |

Table 5: Dataset Library

# Model Design

## Terminologies

### Neural Network

Neural networks are a computer system that is a simulation of how the human brain works. They are a kind of machine learning, which is a kind of Artificial intelligence itself. Given the way they learn, the more data they are given, the “smarter” they will be [7].

### Bag of Words (BOW)

Bag of Words is an encoding method in which the value of each word in the vocabulary is the number of occurrences of that specific word in the training dataset, disregarding the grammar and word orders. As a result of that, the encoded sentence is the numerical-vector representation based on that vocabulary. The encoded sentences are input of the neural network.

### Vocabulary

Consists of words which occurred in the training dataset. The data type of the vocabulary for this project is python dictionary which its key is words, and its value is the number of occurrences of each word withing the training dataset or the index of that word withing the vocabulary dictionary itself.

### Tokenizing

Tokenizing is a process for each sentence within the dataset to prepare them for encoding. In this process, each sentence is sliced into words.

### Encoding

Use the numerical value of each word in the vocabulary dictionary to convert it to a list of numbers corresponding to the value of each word within the vocabulary dictionary. At the same time, each vector is padded to the max length sentence of training dataset by adding zeros to the end of the vector.

### Embedding layer

Takes in encoded sentences and convert them into a pre-determined size numerical vector which is formed considering the position of each word within that specific sentence. It has 2 advantageous:

1. Dimensional reduction. Example: each encoded sentence is a vector of 231 numbers and is reduced to 100 numbers by the embedding layer.
2. Making a semantic numerical presentation for each sentence.

If you use 4 words in two different sequence, you will have 2 different embedded vectors.

### Dense layer

The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most used layer in the models. In the background, the dense layer performs a matrix-vector multiplication.

### Convolutional layer

The convolutional layer in a convolution neural network takes in an input matrix and uses a pre-determined number of kernels with an initial random weight set for each kernel to slide over the matrix. This will form a new convolutional layer called feature maps with the reduced dimension which retains the important information of the original convolutional layer. The feature map is the matrix multiplication of the kernel and the original convolutional layer.

### Dropout layer

The Dropout layer “nullifies the contribution of some neurons towards the next layer and leaves unmodified all others” [8]. It randomly takes off some neurons out of the layer in each iteration of training and prevents all neurons in a layer from synchronously optimizing their weights. This random adaptation prevents all the neurons from converging to the same goal, thus decorrelating the weights.

### Flatten

This layer is used after convolutional layer and before output layer to convert matrices into a vector which is appropriate format for output Dense layer.

### Max pool

It might be used after any convolutional layer. The main purpose of it is that it reduces the size of the convolutional layer by retaining the important information of that Convolutional layer.

### BERT

The BERT layer is a pre-configured layer that can be used in any model. This layer was developed by TensorFlow. First, “each input word is embedded into a token using BERT’s raw embeddings, which is a bag of words of about 30K words.” Second, “the tokenizer adds to each of the encoded tokens a value indicating its sentence index”. Third, the tokenizer adds an input positional index of the token within the training dataset. The following diagram shows an example of this process. [9]

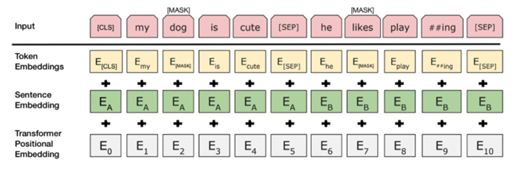


Figure 12: BERT Layer diagram

### Activation Functions

It is a function that is applied on the output of each neurons to form an input data for next layer’s neurons. There are various types of activation functions such as Softmax, Sigmoid, Relu, Logistic, Tanh and Leaky Relu.

### Optimizers

Optimizers in the model use the loss function to determine if the model is changing the weights in the right direction. If the loss increases, then the model is getting worse, and the optimizer would work to correct the wrong direction.

### Threshold

The threshold is referring to the point where sentences are predicted as a requirement only if the predicted probability is more than that specified point.

The following table shows an example of the inputs and outputs of some of the processes described in the terminology section above:

|  |  |
| --- | --- |
| Training Dataset | “The quick brown fox jumped over the lazy dog.”  “The dog and the jumping fox.” |
| Creating Vocabulary | The, quick, brown, fox, jumped, over, lazy, dog, and, jumping |
| Tokenized dataset | |  |  | | --- | --- | | Number of Occurrences | Index | | The: 4  Quick: 1  Brown: 1  Fox: 2  Jumped: 1  Over: 1  Lazy: 1  Dog: 2  And: 1  Jumping: 1 | The: 1  Quick: 2  Brown: 3  Fox: 4  Jumped: 5  Over: 6  Lazy: 7  Dog: 8  And: 9  Jumping: 10 | |
| Encoded Dataset | Choosing the Number of Occurrences method of the value of each token, the following encoded dataset is the result:  [4,1,1,2,1,1,4,1,2],  [4,2,1,4,1,2] |
| Padded Dataset | The max length of the dataset is 9, so padding each sentence to that length with 0 results in the following.  [4,1,1,2,1,1,4,1,2],  [4,2,1,4,1,2,0,0,0] |
| Embedding Layer | Reduce the length of 9 down to 2.  [2.34, 5.32],  [3.5, 4.35] |

Table 6: Threshold dataset and layer examples

## Model Alternatives

### Fully Connected Neural Network (FNN)

The fully connected neural networks consists of input layer, hidden dense layers and output layer. This model uses a Bag of Words method for encoding input sentences into the model.

### Convolutional Neural Network (CNN) Model

This model has convolutional layers in addition to the dense layer. There are different convolutional layer depending on the type of the dataset. For example, for image processing there are two-dimensional convolutional layers, but for the text processing in natural language processing, one-dimension convolutional layers are used.

### BERT Model

It is a neural network used for natural language processing that included BERT layer as an its Embedding layer. It can be used with either convolutional neural networks or regular fully connected neural networks.

## Model Architectures

### Fully Connected Neural Network (FNN) Model

The FNN in this project uses a Bag of Words method of encoding sentences. There are 2 hidden layers consisting of 80 and 50 nodes or neurons respectively. The output layer is a one node layer which generate a probability of being requirement for any sentences.

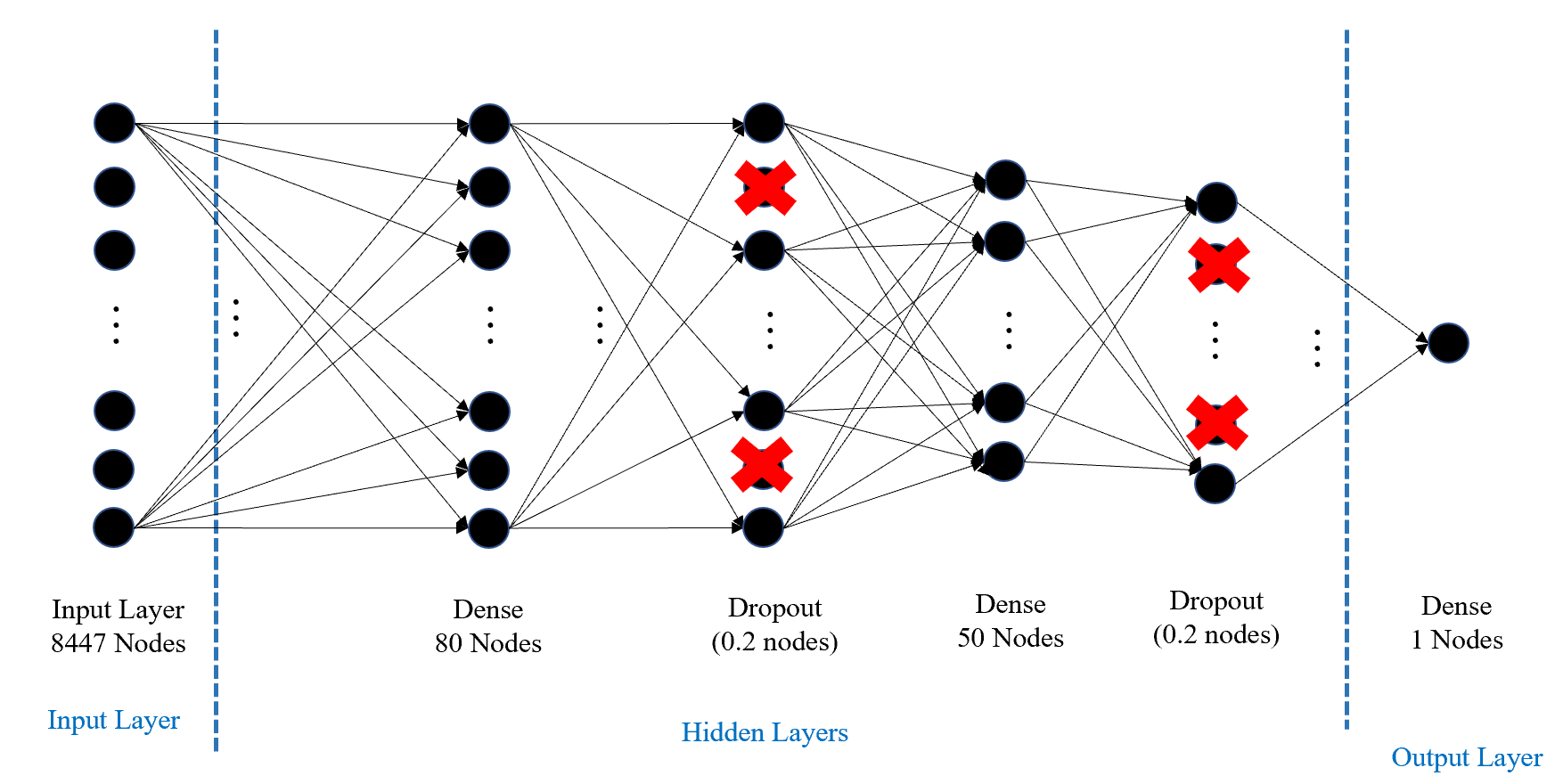


Figure 13: FNN diagram

### Convolutional Neural Network (CNN) Model

In the CNN model, there is an Embedding layer of size 100 which reduces the dimension of the encoded sentences of size 231 to a vector of size 100. Next layer is a one-dimensional convolutional layer with 32 feature maps layers. After that, a spatial one-dimensional dropout layer, Maxpool convolutional layer to reduce the size, the Flatten layer to make a 1D vector from convolutional layer output, Dense layer of size 10 nodes and finally a dropout layer before output.

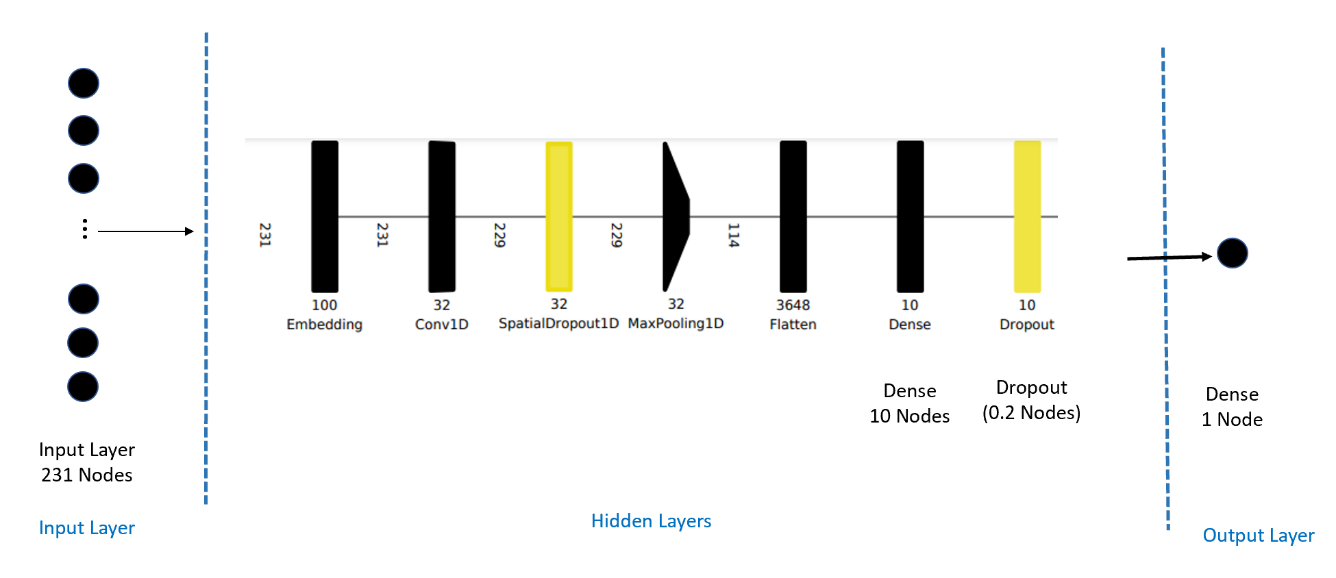


Figure 14: CNN diagram

### BERT Model

In the BERT model, a use pre-configured BERT layer as an embedding layer is used. The output of the Embedding layer is a vector of size 100 for each encoded sentence. After that, there are 2 hidden layer of size 64 and 32 respectively. After each hidden dense layer, there is a dropout layer which randomly nullifies 0.2 precents of nodes or neuron of each layer in each epoch of the training the mode.

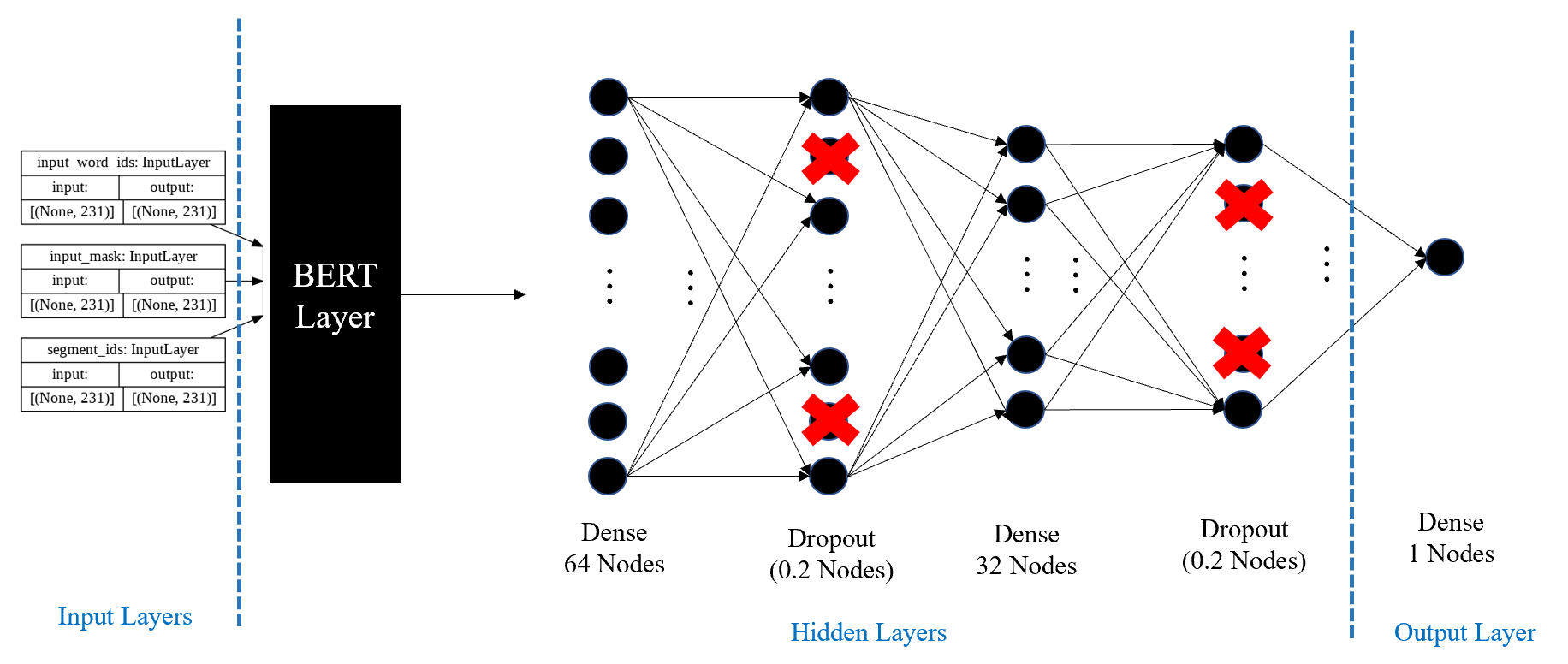


Figure 15: BERT model diagram

# Implementation

## Base Models

### FNN Model

The initial FNN model that was the developed did not have dropout layers and used ReLU activation function for the output of the hidden layers. The test accuracy of the model was 71% with the F1 score of 0.56 and MCC score of 0.29. After running the model with different hyper-parameters such as number of layers, number of nodes in each layer, different activation functions, different threshold and adding dropout layer, the best result was achieved by the one with two dropout layers, one after each hidden dense layer and activation function of SoftMax for each hidden layer. The test accuracy of this kind of FNN has increased to 77% with F1 score of 0.58 and MCC score of 0.26. Because the MCC and F1 Score are almost the same as the initial models’ and the test accuracy of the modified model is noticeably higher than 71%, the final model used was the modified model with the higher test accuracy. In the training phase of the model, the model shuffles the training dataset in each iteration and uses a different validation dataset, which both factors increase the generalizability of the model.

### CNN Model

The initial model that was developed of the CNN model included an Embedding layer of size 100, one-dimension convolutional layer, one Maxpool layer and one dense layer. The test accuracy achieved using that model was 76% along with the F1 and MCC score of 0.49 and 0.18 respectively. After modifying the hyperparameters such as number of layers, number of nodes in each layer, different activation functions, different threshold and adding dropout layer, the best result was achieved by the one with one spatial dropout layer after the convolutional layer and one dropout layer after the dense layer with the same activation functions and layers’ size as the initial CNN model. The accuracy remains almost the same as 76% but the F1 and MCC scores has increased noticeably to 0.63 and 0.38 respectively. In the training process of the model, in each iteration of training after each forward and backward propagation, it shuffles the training dataset and uses different validation dataset. These factors increase the generalizability of the model.

### BERT Model

Because of the high computational cost of the BERT model which has hundred and nine million parameters for training, the team were not able to examine different hyper parameters. For training the BERT model, the ARGO account of the George Mason University was used, and the time of the training process was 48 hours. This model includes pretrained and configured BERT layer, developed by TensorFlow, which is a sophisticated embedding layer [10]. In addition to the BERT layer, this model includes two dropout layers after each dense layer. The activation function of ReLU is used for each dense layer. The test accuracy of the model is 81% with having F1 and MCC scores of 0.55 and 0.2 respectively.

## Model Combinations

In order to determine if combining models would produce better results, several combinations of model outputs were combined to get a new set of predictions for the sentences. They are described in the following sections.

### One out of Three

This set of predictions predicts that a sentence is a requirement if at least one of the three base models (FNN, CNN, or BERT) predicts that the sentence is a requirement.

### Two out of Three

This set of predictions predicts that a sentence is a requirement if at least two of the three base models (FNN, CNN, or BERT) predicts that the sentence is a requirement.

### Three out of Three

This set of predictions predicts that a sentence is a requirement if all three base models (FNN, CNN, and BERT) predicts that the sentence is a requirement.

### BERT and CNN Agree

This set of predictions predicts that a sentence is a requirement if both the BERT model and the CNN model predict that the sentence is a requirement.

### BERT and FNN Agree

This set of predictions predicts that a sentence is a requirement if both the BERT model and the FNN model predict that the sentence is a requirement.

## Results

After all the data was labeled, the dataset was split into 94% for training data and 6% Test data. This test data consisted of two original requirements from the original document dataset. In addition, a threshold for all the models needed to be determined. This threshold is the value where if the predicted probability is greater than that value, then it is identified as a requirement and below, it is not. For all models, a threshold of 0.5 was used. Changing this value would change the results. The following results are all calculated using the test data.

### Confusion Matrix

A confusion matrix was first created to evaluate the performance of the models. It contains the percentage of true positives, true negatives, false positives, and false negatives. These values are used to calculate other performance metrics. True positives (TP) are the percentage of correctly identified requirements by the model. Similarly, true negatives (TN) are the percentage of non-requirement sentences that are correctly identified by the model. False positives (FP) are the percentage of requirements that are incorrectly identified as requirements by the model. False negatives (FN) are the percentage of requirements that are incorrectly labeled as non-requirements by the model. A perfect model would have a true positive and true negative percentage that add to 1 while false positive and false negative are 0.

The following shows how the confusion matrix is set up. The left side shows the actual status of the requirements while the top row shows the prediction by the model.

|  |  |  |
| --- | --- | --- |
|  | Model Predicts Requirement | Model Predicts Non-Requirement |
| Requirement | TP | FN |
| Non-Requirement | FP | TN |

Table 7: Confusion Matrix example

The confusion matrix for the Fully Connected Neural Network model is as follows:

|  |  |  |
| --- | --- | --- |
| Fully Connected Neural Network Model | Model Predicts Requirement | Model Predicts Non-Requirement |
| Requirement | 0.15 | 0.10 |
| Non-Requirement | 0.13 | 0.62 |

Table 8: Confusion Matrix results for FNN model

The confusion matrix for the Convolutional model is as follows:

|  |  |  |
| --- | --- | --- |
| Convolutional Model | Model Predicts Requirement | Model Predicts Non-Requirement |
| Requirement | 0.22 | 0.03 |
| Non-Requirement | 0.21 | 0.54 |

Table 9: Confusion Matrix results for CNN model

The confusion matrix for the Semantic model that implements a BERT Layer is as follows:

|  |  |  |
| --- | --- | --- |
| Semantic BERT Model | Model Predicts Requirement | Model Predicts Non-Requirement |
| Requirement | 0.17 | 0.08 |
| Non-Requirement | 0.17 | 0.59 |

Table 10: Confusion Matrix results for BERT layer

The confusion matrix for the model that predicts a requirement if at least one model predicts a requirement is as follows:

|  |  |  |
| --- | --- | --- |
| 1 Out of 3 Model | Model Predicts Requirement | Model Predicts Non-Requirement |
| Requirement | 0.23 | 0.01 |
| Non-Requirement | 0.29 | 0.46 |

Table 11: Confusion Matrix results for 1 Out of 3 model

The confusion matrix for the model that predicts a requirement if at least two models predicts a requirement is as follows:

|  |  |  |
| --- | --- | --- |
| 2 Out of 3 Model | Model Predicts Requirement | Model Predicts Non-Requirement |
| Requirement | 0.18 | 0.07 |
| Non-Requirement | 0.14 | 0.61 |

Table 12: Confusion Matrix results for 2 Out of 3 model

The confusion matrix for the model that predicts a requirement if all three models predicts a requirement is as follows:

|  |  |  |
| --- | --- | --- |
| 3 Out of 3 Model | Model Predicts Requirement | Model Predicts Non-Requirement |
| Requirement | 0.12 | 0.12 |
| Non-Requirement | 0.07 | 0.68 |

Table 13: Confusion Matrix results for 3 Out of 3 model

The confusion matrix for the model that predicts a requirement if the CNN and BERT model agree predicts a requirement is as follows:

|  |  |  |
| --- | --- | --- |
| BERT & CNN Agree | Model Predicts Requirement | Model Predicts Non-Requirement |
| Requirement | 0.16 | 0.09 |
| Non-Requirement | 0.11 | 0.64 |

Table 14: Confusion Matrix results for CNN & BERT Agree model

The confusion matrix for the model that predicts a requirement if the FNN and BERT model agree predicts a requirement is as follows:

|  |  |  |
| --- | --- | --- |
| BERT & FNN Agree | Model Predicts Requirement | Model Predicts Non-Requirement |
| Requirement | 0.13 | 0.12 |
| Non-Requirement | 0.08 | 0.67 |

Table 15: Confusion Matrix results for FNN & BERT Agree model

### Model Accuracy

Accuracy is one way to measure the performance of a Machine Learning Neural Network. It is a measure of the total number correct predictions over the total of all predictions. Each of the models have been evaluated for their accuracy. This is calculated by the following equation:

With the threshold for each of the models at 0.5, the accuracy is as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| FNN | CNN | BERT | 1 Out of 3 | 2 out of 3 | 3 Out of 3 | BERT&CNN Agree | Bert & FNN agree |
| 0.77 | 0.76 | 0.76 | 0.69 | 0.79 | 0.81 | 0.79 | 0.80 |

Table 16: Accuracy results for various models

The accuracies of all the models are not very different. The highest accuracies are found with the combinations of the three base models.

### Sensitivity and Specificity

Sensitivity is the true positive rate while specificity is the true negative rate. High sensitivity would mean the model predicts the true requirements well. High specificity would mean that the model predicts the true negatives well. The best case is where both are 1. This would mean that all the true negatives were identified, and all of the true positives were identified. However, there may be trade-offs that need to be made between them if there is not one model that has both high specificity and sensitivity. The equations are as follows:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | FNN | CNN | BERT | 1 Out of 3 | 2 out of 3 | 3 Out of 3 | BERT&CNN Agree | Bert & FNN agree |
| Sensitivity | 0.61 | 0.88 | 0.68 | 0.94 | 0.73 | 0.50 | 0.63 | 0.52 |
| Specificity | 0.83 | 0.72 | 0.78 | 0.61 | 0.81 | 0.91 | 0.85 | 0.90 |

Table 17: Sensitivity and Specificity Results

The highest sensitivity is shown with the model that identifies a requirement if at least one of the three models identifies it as a requirement. However, this model has the lowest specificity. This is because the false positive rate was higher in order to get the higher sensitivity. The highest specificity is shown with the model that identifies a requirement if at least one of the three models identifies it as a requirement and the model where the semantic model and FNN model agree. In both, the sensitivity is very low. This tradeoff between the sensitivity and specificity can be seen in the following plot.

Figure 16: Sensitivity vs Specificity graph

### F1 Score

The F1 Score is another method of measuring the accuracy of the models. It is the harmonic mean of the precision and recall. Recall is the same as sensitivity, and precision in this case is the number of true positives over the number of positively identified requirements. The formula is as follows:

Using the classification rates, the following F1 scores were obtained:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| FNN | CNN | BERT | 1 Out of 3 | 2 out of 3 | 3 Out of 3 | BERT&CNN Agree | Bert & FNN agree |
| 0.58 | 0.65 | 0.58 | 0.61 | 0.64 | 0.56 | 0.61 | 0.57 |

Table 18: F1 Score results of various models

However, there may be problems with using the F1 score for evaluating the model performance when there is a biased dataset, like the one used in these models with only 25% requirements, because it is independent of the number of true negatives [11]. The F1 score does not account for the correctly identified non-requirements in the sample, which makes up a large portion of the sentences.

### Mathews Correlation Coefficient

Mathews Correlation Coefficient is an alternative to the F1 score of the accuracy of model prediction. It may be a better representation of the model performance than the F1 score because it is not affected by the bias of the dataset. It only produces a high score when there are a high number of true positives and true negatives [11]. The equation for calculating the MCC is the following:

The MCC for each model was obtained:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| FNN | CNN | BERT | 1 Out of 3 | 2 out of 3 | 3 Out of 3 | BERT&CNN Agree | Bert & FNN agree |
| 0.26 | 0.39 | 0.28 | 0.42 | 0.32 | 0.22 | 0.27 | 0.22 |

Table 19: MCC results of various models

### Bayes Probability

Using Bayes’ theorem, the conditional probability can be used to determine the probability that a sentence is a requirement given that the model predicted it is a requirement. Similarly, the probability that a sentence is not a requirement given that the model predicted that it was not a requirement can also be found. The conditional probability is shown in the following equations:

The results of these are as such:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | FNN | CNN | BERT | 1 Out of 3 | 2 out of 3 | 3 Out of 3 | BERT&CNN Agree | Bert & FNN agree |
| Bayes Probability Requirement | 0.54 | 0.51 | 0.51 | 0.45 | 0.56 | 0.64 | 0.58 | 0.63 |
| Bayes Probability Not a Requirement | 0.58 | 0.34 | 0.55 | 0.23 | 0.50 | 0.62 | 0.57 | 0.62 |

Table 20: Bayes probability results for various models

### Summary of Results

All the results from the above discussion are summarized in the following table.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | FNN | CNN | BERT | 1 Out of 3 | 2 out of 3 | 3 Out of 3 | BERT&CNN Agree | Bert & BOW agree |
| TP | 0.15 | 0.22 | 0.17 | 0.23 | 0.18 | 0.12 | 0.16 | 0.13 |
| TN | 0.62 | 0.54 | 0.59 | 0.46 | 0.61 | 0.68 | 0.64 | 0.67 |
| FP | 0.13 | 0.21 | 0.17 | 0.29 | 0.14 | 0.07 | 0.11 | 0.08 |
| FN | 0.10 | 0.03 | 0.08 | 0.01 | 0.07 | 0.12 | 0.09 | 0.12 |
| Accuracy | 0.77 | 0.76 | 0.76 | 0.69 | 0.79 | 0.81 | 0.79 | 0.80 |
| F1 | 0.58 | 0.65 | 0.58 | 0.61 | 0.64 | 0.56 | 0.61 | 0.57 |
| MCC | 0.26 | 0.39 | 0.28 | 0.42 | 0.32 | 0.22 | 0.27 | 0.22 |
| Sensitivity | 0.61 | 0.88 | 0.68 | 0.94 | 0.73 | 0.50 | 0.63 | 0.52 |
| Specificity | 0.83 | 0.72 | 0.78 | 0.61 | 0.81 | 0.91 | 0.85 | 0.90 |
| Bayes Probability Requirement | 0.54 | 0.51 | 0.51 | 0.45 | 0.56 | 0.64 | 0.58 | 0.63 |
| Bayes Probability Not a Requirement | 0.58 | 0.34 | 0.55 | 0.23 | 0.50 | 0.62 | 0.57 | 0.62 |

Table 21: Overall results of various models

# Analysis of Strategy Alternatives

Different strategies may meet the objectives of different reviewers. If the goal is to reduce the workload on reviewing a document but finding all true requirements is less important, then the models with a lower MCC and highest percentage of true requirements given to the reviewer should be selected. If the objective is to find the most true requirements, but workload is not as important, then a higher MCC and lower false negative rate should be found. As the false negative rate decreases, the false positive rate increases. Consequently, as the false positive rate increases in the models with a lower false negative rate, there is an increase in the number of sentences to be reviewed. This is the tradeoff between the total number of requirements to review and the missed false negatives not reviewed.

## Strategy 1: Greatest Percent Requirements Identified

Choosing the model that identifies a requirement as a requirement if at least one of the three base models identifies it as a requirement would result in identifying the highest percentage of true requirements, but the need to review the most sentences. However, even with this model, based on the test data, the reviewer would need to review just half of the sentences from the entire document reducing the workload by 50% and will receive 93% of the true requirements.

## Strategy 2: Least Sentences to Review

Choosing a model that identifies the fewest sentences of a document to review but the with the least identified requirement would result in choosing the model that identifies a requirement if at least two of the base models identifies it as a requirement. With this model, there is a higher error rate (false negatives of all true requirement) of about 50%, but only about 20% of the original requirements document would need to be reviewed reducing the workload by 80%.

## Strategy 3: Balance Number of Sentences to Review and Error

In order to balance the number of sentences to review and the error rate, then a model somewhere in between the previous models should be chosen. The models follow a linear pattern with a slope of 1.39. So, in general, based on the trend of these models, increasing the number of sentences to review by 10% increases the number of requirements found by 13.9%.

The following plot shows these findings.

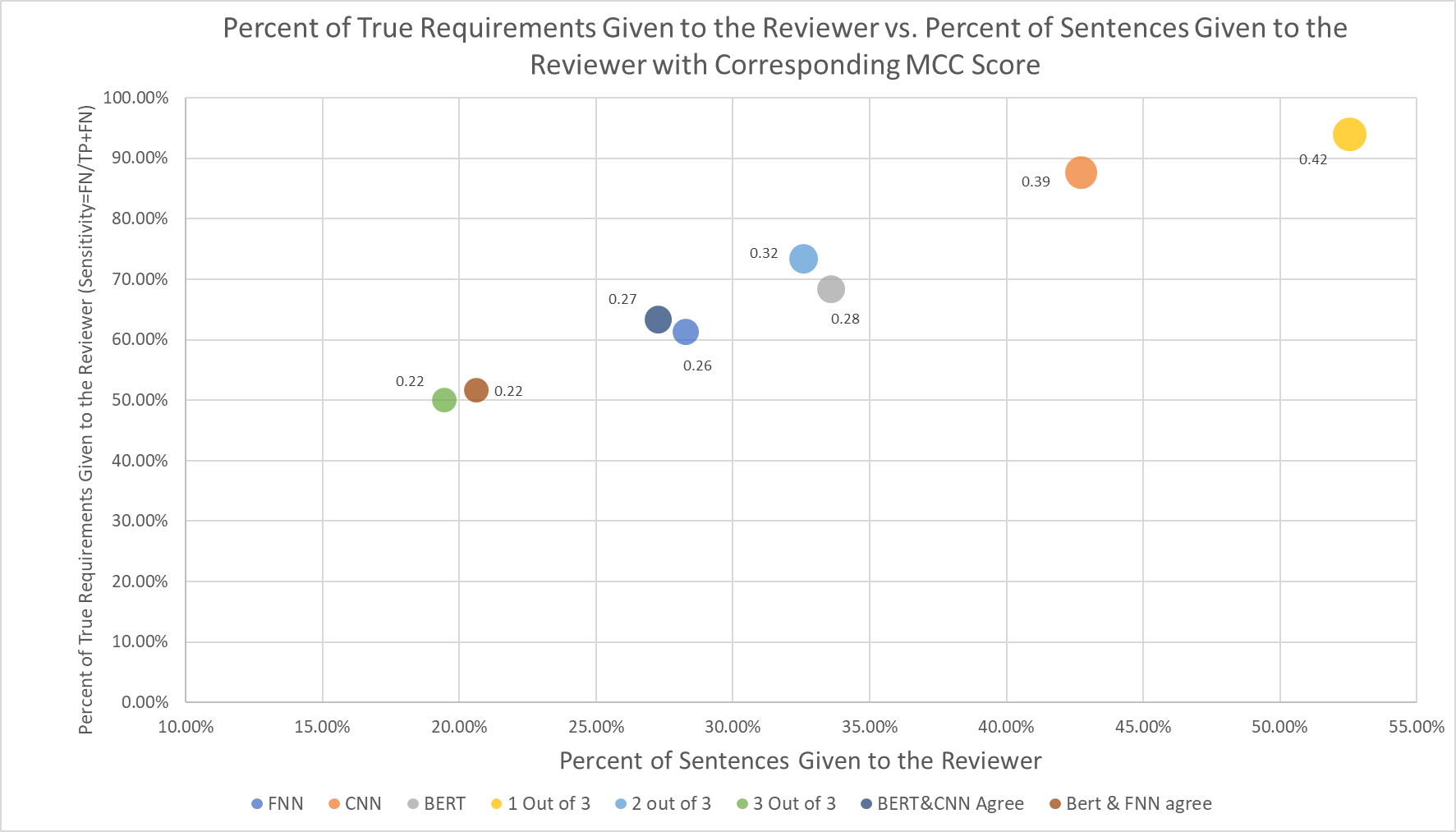
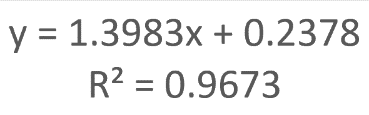


Figure 17: Graph of Sentences given to reviewer vs true sentences identified

## Improvement

An improvement in the models would show a decrease the slope of this line and shift it to the up and to the left. This would the percent of requirement identified while decreasing the number of sentences needed to review. To do this, one methods of improvement may be adding more data to the training data.

# Prototype

A prototype of the requirements extractor has been developed to allow a user to classify requirements from the proposal document. The application has the following elements; File Explorer (to select file), User Interface to view basic elements of AI classification and output, and a report output from the file.

The file explorer allows a user to select a Word or PDF document that they want to be read and processed by the AI algorithm. This utilizes the file explorer of the native operating system. When selected, the file will be ingested into the algorithm. Upon processing, stats on the document will be displayed; how many requirements were found and confidence of different algorithms. A more detailed report will be generated with the algorithm’s classifications and predictions. The user will be able to view all the requirements found and their respective confidence levels.

In addition, for each set of predictions the identified requirements will be highlighted in yellow in different tabs of the Excel. This way, a reviewer could choose the prediction method they wand and see which requirements were positive predictions by only looking at the highlighted sentences. By showing the requirements this way, the reviewer also gets the context of the sentence by seeing the sentences before and after the identified requirement. An example of the highlighting is seen in Table 23.

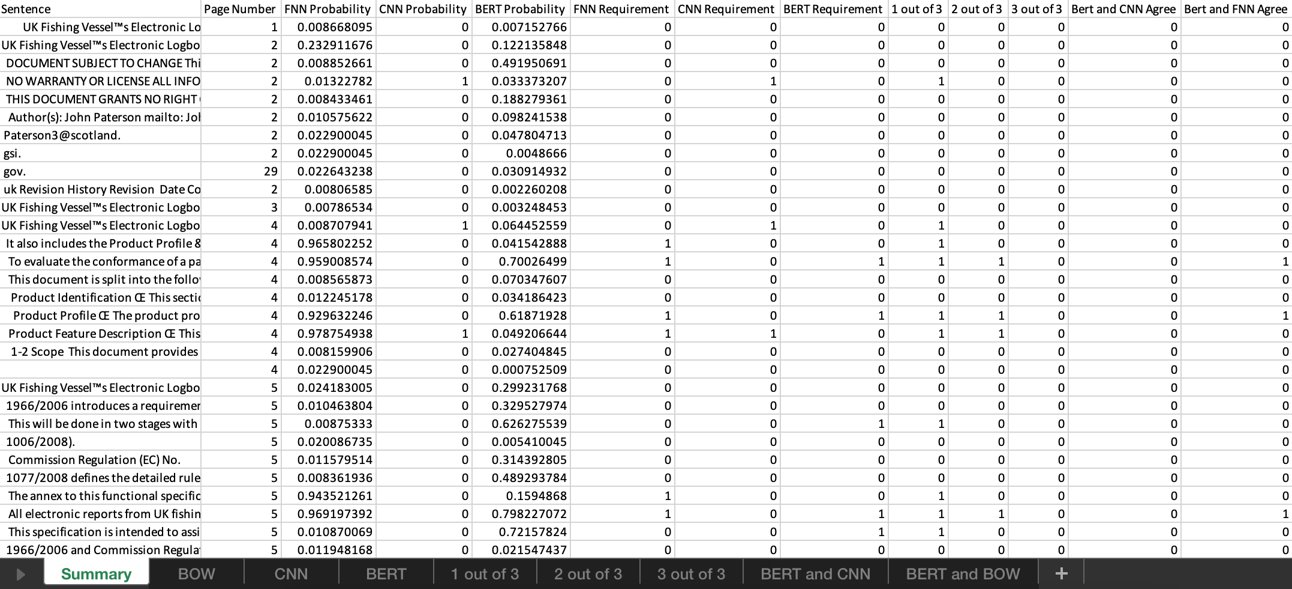


Table 22: Sample Algorithm Output

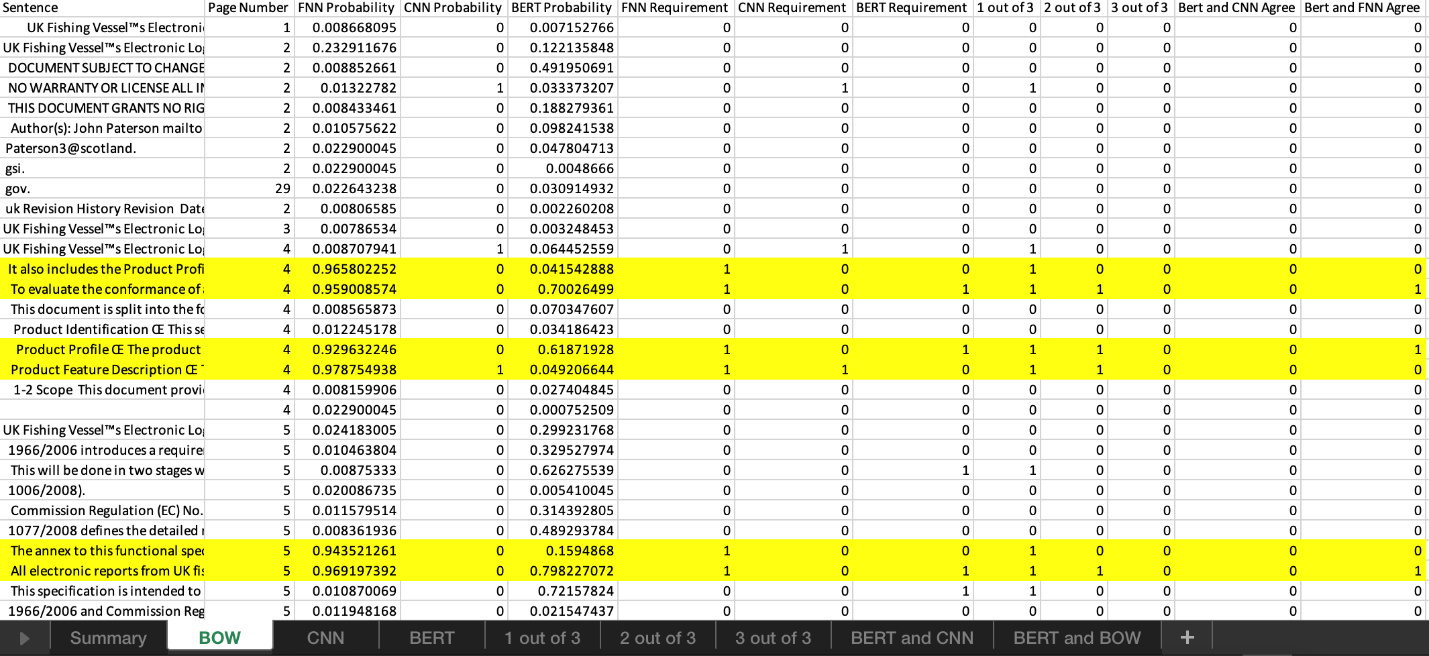


Table 23: Sample Algorithm Output with Highlights

# Conclusion

In this paper, historical analysis of AI/ML integration into systems engineering was outlined, a use-case for NLP integration into requirements engineering was detailed, an NLP-based system design was described, and analyses of the results were performed using varied statistical techniques. The reduction of review load of requirements on the requirements engineer is estimated between 50-80%, depending on the user’s comfort level with true requirement identification. A more time-focused stakeholder may opt to cut the review load by 80% and would accept the 50% true requirement identification hit. Or a more quality-focused stakeholder may opt to cut the review load by 50% and still identify 90% true requirements.

# Future Work

Future work has been identified as below:

* Increase the accuracy of the models
  + Increasing the accuracy may be accomplished by adding stemming to the preprocessing, adding more data to the training dataset, or developing a custom BERT layer appropriate for MITRE.
* Create new development that applies to more areas of requirements analysis
  + New development may include creating a model that can identify the subject of the requirements or creating a model that can rewrite/correct a requirement in one document into one standard and correct format. In addition, training models on requirement documents of different industries individually may produce different trained models that perform better in their industries.
* Improve the testing of the models
  + Testing of the models may be improved by running multiple replications of each model to get a mean and standard deviation of the measurement scores or by testing with multiple test datasets.

# Challenges

Challenges that have been identified in this development process include:

* Data collection
* Data labeling
* Difficulty of slicing documents into sentences
* Differing perceptions of identifying requirements and non-requirements
* High Computation Cost
* Increasing Test Accuracy of Models

# References

|  |  |
| --- | --- |
| [1] | The MITRE Corporation, "The Evolution of Systems Engineering," MITRE, 2016. |
| [2] | M. Petrotta and T. Peterson, "Implementing Augmented Intelligence in Systems Engineering," in *NDIA Ground Veh. Sys. Eng. and Tech. Symp.*, Novi, 2018. |
| [3] | K. Verma and A. Kass, "Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Docuements," Accenture Technology Labs, San Jose, 2008. |
| [4] | G. Lami, "QuARS: A Tool for Analyzing Requirements," in *Software Engineering Measurement and Analysis Initiative*, 2005. |
| [5] | Respect-IT, "A KAOS Tutorial," 2007. [Online]. Available: http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf. [Accessed Feb 2021]. |
| [6] | National Research Council of Italy, "Natural Language Requirements Dataset," [Online]. Available: http://nlreqdataset.isti.cnr.it/. |
| [7] | J. Frankenfield, "Artificial Neural Network (ANN)," Investopedia, 8 Aug 2020. [Online]. Available: https://www.investopedia.com/terms/a/artificial-neural-networks-ann.asp#:~:text=An%20artificial%20neural%20network%20%28ANN%29%20is%20the%20piece,impossible%20or%20difficult%20by%20human%20or%20statistical%20standards. [Accessed Feb 2021]. |
| [8] | Baeldung, "How ReLU and Dropout Layers Work in CNNs," 13 Aug 2020. [Online]. Available: https://www.baeldung.com/cs/ml-relu-dropout-layers#:~:text=The%20Dropout%20layer%20is%20a%20mask%20that%20nullifies,in%20which%20case%20it%20nullifies%20some%20hidden%20neurons. [Accessed Feb 2021]. |
| [9] | M. Cohen-Yashur, "NLU for everyone with BERT," Medium, 8 Apr 2020. [Online]. Available: https://medium.com/swlh/nlu-for-everyone-with-bert-7bedaa609a61 . [Accessed Feb 2021]. |
| [10] | TensorFlowHub, "Bidirectional Encoder Representations from Transformers (BERT)," 4 May 2021. [Online]. Available: https://tfhub.dev/tensorflow/bert\_en\_uncased\_L-12\_H-768\_A-12/2 . [Accessed Feb 2021]. |
| [11] | D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics,* vol. 21, no. 6, 6 Jan 2020. |
| [12] | "Net2Vis," Viscom, [Online]. Available: https://viscom.net2vis.uni-ulm.de/OG1Br2BAkYSwwrV6CADl4X5EfErFjUzvuUwXWDdLbdsIXNhb9L. [Accessed 1 May 2021]. |
| [13] | J. Brownlee, Deep Learning for Natural Language Processing. |