Artificial Intelligence for Systems Engineering: Requirements Identification

Megan Taylor   
Department of Systems Engineering and Operations Research  
George Mason University  
Fairfax, VA, United States  
[mtaylo39@gmu.edu](mailto:mtaylo39@gmu.edu)

Seyed-Omid Beheshti-Shirazi  
Department of Systems Engineering and Operations Research  
George Mason University  
Fairfax, VA, United States  
[sbehesh5@gmu.edu](mailto:sbehesh5@gmu.edu)

Michael Tzimourakas  
Department of Systems Engineering and Operations Research  
George Mason University  
Fairfax, VA, United States  
[mtzimour@gmu.edu](mailto:mtzimour@gmu.edu)

Alex Zepka  
Department of Systems Engineering and Operations Research  
George Mason University  
Fairfax, VA, United States  
[azepka@gmu.edu](mailto:azepka@gmu.edu)

*Abstract*—MITRE has tasked the GMU Systems Engineering and Operations Research (SEOR) department with identifying opportunities to incorporate Artificial Intelligence (AI) and Machine Learning (ML) into the systems engineering process. This paper explores the potential of improving the systems engineering process in areas of requirements identification by integrating a natural language processing (NLP) Machine Learning algorithm to process and extract requirements from project proposals as well as proposal submissions. The algorithm will also identify stated capabilities that should be traced back to requirements.

Several different text classification algorithms were analyzed for use, such as a Fully Connected Neural Network, Convolutional Neural Network, and a Semantic Neural Network that implements a BERT layer developed by Tensorflow. Upon completing training, the NLP algorithm will process proposal documents and responses and predict whether the sentence is a requirement or not. Combinations of the model predictions were analyzed in addition to the individual model predictions.   For testing the models, the algorithm was fed the test data, and the tagged requirements were compared to the actual label. The model accuracy, F1 score, and MCC were calculated. Based on the results of the various model predictions, a tradeoff between the number of true requirements elicited from the models and the percent of the entire document to be reviewed was found. This tradeoff is discussed in the paper and used to create three strategies to use the AI.

Keywords—Natural Language Processing, NLP, Machine Learning, ML, Artificial Intelligence, AI, Systems Engineering, Requirements

# Introduction

The proposal review process is a necessary and time-consuming process from the perspective of the solicitors that must read through 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 understaffed, 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.

A solution to minimize errors and intensive manual efforts in the requirement process is to utilize a NLP (Natural Language Processing) ML (Machine Learning) Algorithm to process and extract requirements from project proposals as well as proposal submissions. The requirements will be written and presented into a usable form that assists the user in determining the winner of such contract solicitations.

For this project, the NLP algorithm allows the user to independently review requirements documents. The algorithm shall ingest proposal documents (PDF, docx) and strip out each sentence and write it to an Excel file. This process will allow the user to skip the time required to manually extract requirements from the proposal document and begin assessing immediately the requirements of the proposals and their responses. The system will also identify stated capabilities that should be traced back to requirements.

The project sponsor, MITRE, has also tasked the SEOR department with identifying opportunities for AI/ML integration in the various phases of the systems engineering process. This task was completed with a literature overview of past examples of AI/ML in the systems engineering process.

# Literature Survey

A literature survey of existing AI/ML integration efforts in the systems engineering process was performed. A handful of systems and papers were found over the past 20 years. In 2005, a paper was written, “QuARS: A Tool for Analyzing Requirements,” which details a software system that detects linguistic inaccuracies or defects in requirements language [1]. In 2008, a paper was written, “Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents,” which details a software system (RAT) that performs a semantic analysis over a requirements document to also identify linguistic inaccuracies in order to improve requirements language [2]. KaOS, a commercial software product, was designed to assist the engineer in requirements derivation by creating a requirements model with which to base their glossary on and to formally write requirements [3].

Outside of requirements, a 2018 paper describes tracing requirements changes to SWAP-C (size, weight, power, and cooling) alterations [4]. The same paper also proposes using intelligent modeling to apply fault isolation strategically to system areas most prone to faults. Based on these historical examples, it is clear requirements engineering is a well-identified opportunity for AI/ML integration. A reason for this may be that requirements engineering is a very data intensive area of systems engineering, which is an ideal opportunity for AI/ML integration.

# System Design

## Problem and Need

In discussions with MITRE, the team distilled the problem and need statements. Table 1 lists the problem and needs:

**Table 1: Problem and Needs Statements.**

| Problem | Need |
| --- | --- |
| AI has been developed since the 1950’s, yet there is a gap in the systems engineering field in implementing AI | Identify an inefficiency in the practice of the systems engineering development process that artificial intelligence (AI) can assist |
| 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. |

## Concept of Operations (ConOps)

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. For the user to tailor the algorithm for their respective use case, the AI must be retrained with a dataset 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. The ConOps for the NLP system is shown in Figure 1.



**Figure 1: System ConOps**

## Requirements Definition

The requirements for the project were created with help from the customer, MITRE. These requirements tie to the problem and needs statement, or the program objectives. Lower-level requirements for the system were derived but are not listed in this paper. The top-level mission requirements are:

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

### The system shall have the ability to be trained on labeled data.

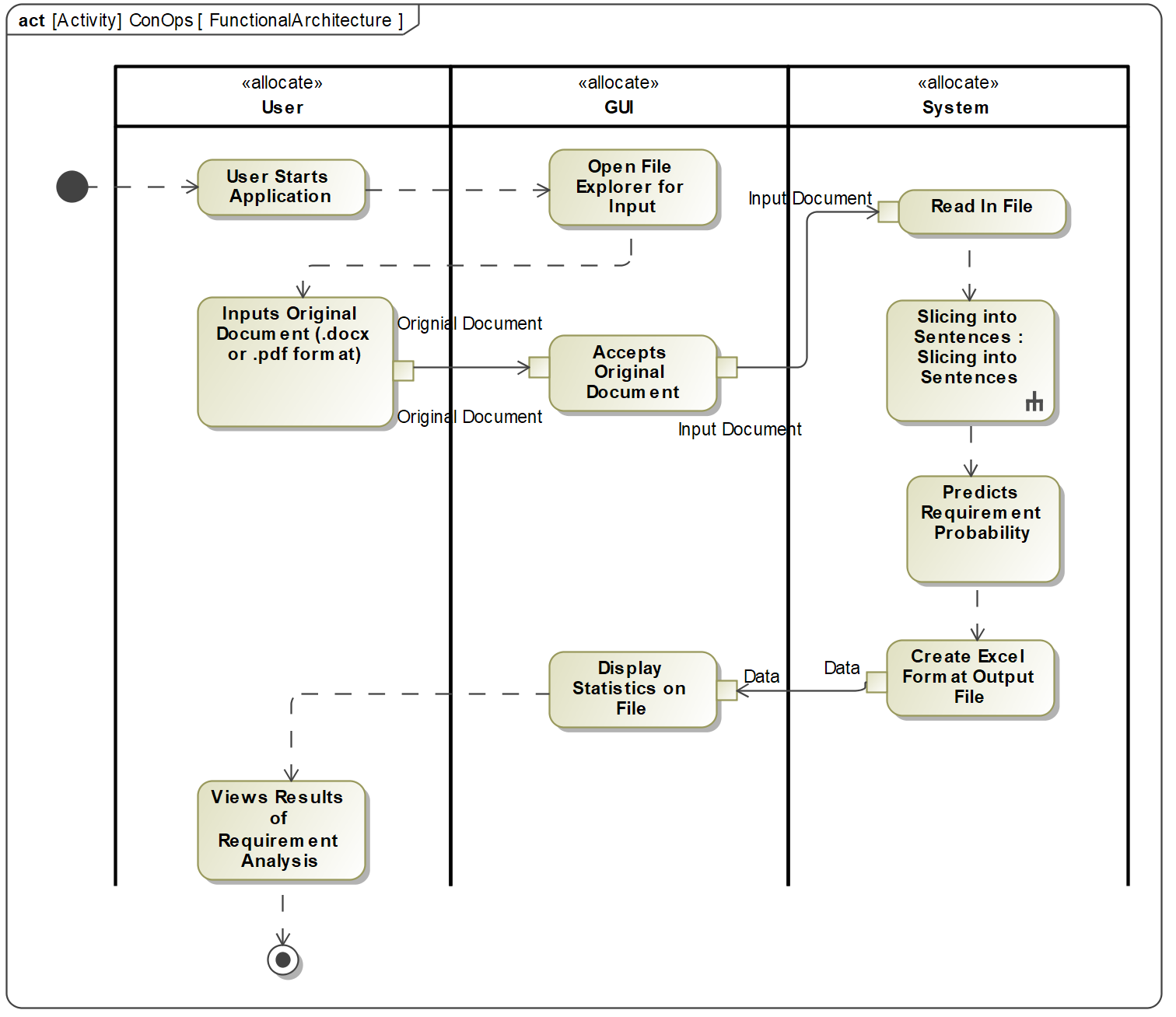
### The system shall predict the probability of being a requirement for each sentence with an MCC Score of at least 0.4.

### The system shall accept an original document from the user (PDF, DOCX).

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

## Functional Architecture

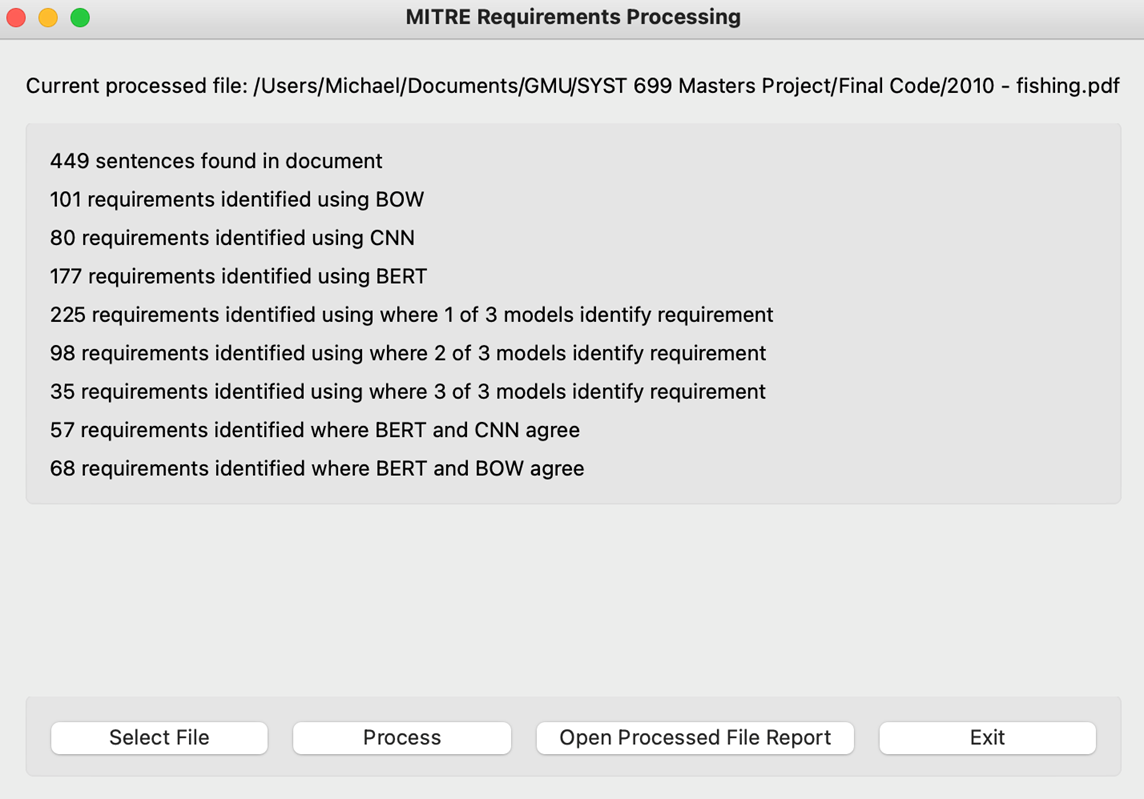
The functional architecture was derived after the requirements definition. The architecture is shown in Figure 2. The use of the system starts when the user starts the application. A Graphical User Interface (GUI) opens with a file explorer for the user to input the original document. Then, the user 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.



**Figure 2: Functional Architecture.**

## System Use – Graphical User Interface

To provide a front end to the developed system, a Graphical User Interface was developed. It contains several 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. A concept graphic for the GUI is shown in Figure 3:



**Figure 3: GUI Prototype.**

## Methodology for Developing a Trained AI Algorithm

The steps for the methodology to develop a trained AI algorithm are shown in the Figure 4. 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.

## Objectives

The objectives governing the development of the AI prototype are:

### Train the AI model.

### Take in a Docx or PDF original 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

To develop the models, assumptions had to be made with the development and the training. They are as follows:

### 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.

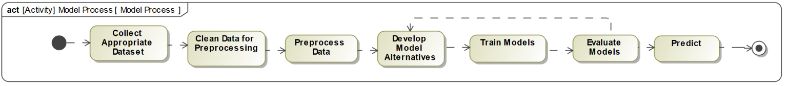
### The models are not able to differentiate different subjects of requirements, just identify that they are requirements.

## Dataset 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 [5]. Each document contains a system and its respective requirements.

## Data Preprocessing

To create the training and testing data, requirements document sentences must be sorted and flagged by a human. Each sentence from the requirements document was extracted and exported into an Excel file with 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.



**Figure 4: Methodology to Develop AI Algorithm**

Table 2 shows a summary of the documents and the number of labeled sentences with the proportion of labeled requirements and non-requirements.

**Table 2: Dataset Overview.**

| 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​ |

## Implementation of Model Architectures

Three different ML algorithms were implemented and trained. Multiple were developed to determine if one performs better than the others. Each one was trained with the training data to allow the models to “learn” what a requirement is. They include a Fully Connected Neural Network (FNN), a Convolutional Neural Network (CNN), and a BERT model. They are described as follows.

### Fully Connected Neural Network (FNN) Model

Multiple combinations of hyperparameters were tested such as the number of layers, number of nodes in each layer, different activation and optimization functions, different threshold and adding dropout layers. The developed FNN uses a Bag of Words method of encoding sentences. There are 4 hidden layers consisting of 80 and 50 nodes or neurons respectively with a dropout layer after each dense layer that randomly nullifies 20% of the nodes in the layer. The activation function used for the hidden layers and the output layer was SoftMax and sigmoid, respectively. The optimizer used was Adam. The best number of epochs was found to be 30. The output layer is a one node layer which generate a probability of being requirement for any sentences.

In the training phase of the model, the model shuffles the training dataset in each iteration and uses a different validation dataset Both of these factors increase the generalizability of the model.

### Convolutional Neural Network (CNN) Model

Different combinations of hyperparameters were also tested with the CNN model such as number of layers, number of nodes in each layer, different activation functions, different threshold and adding dropout layer. In the developed 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. The next layer is a one-dimensional convolutional layer with 32 feature maps layers. After that, a spatial one-dimensional dropout layer, a Maxpool convolutional layer to reduce the size, a Flatten layer to make a 1D vector from convolutional layer output, a dense layer of size 10 nodes and finally a dropout layer before the output with one node. The activation function used for the hidden layers and the output layers was relu and sigmoid, respectively. The optimizer used was Adam. The best number of epochs was found to be 40.

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

Lastly, a model that implements a pretrained and configured BERT layer was developed because it implements a sophisticated embedding layer [7]. 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 20% of nodes of each layer in each epoch of the training the mode. The output is a dense layer of one node. Three epochs were run to train this model.

There were over one-hundred and nine million parameters for training, causing a high computational cost. Training using an ARGO account of George Mason University took approximately 48 hours. Because of this, the team was not able to examine different hyperparameters.

## Model Combinations

To determine if combining the predictions of the three models would produce better results than the individual models alone, 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

## Model Testing

After all the data was labeled, the dataset was split into 94% for training data and 6% Test data. In addition, a threshold for all the models needed to be determined. This threshold is the value of the prediction where if the prediction is greater than the 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. Table 3 shows the confusion matrix results. The results themselves are not used to determine the model performance, but they are used in further calculations.

**Table 3: Confusion Matrix Results.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |

## Model Accuracy

Accuracy is one way to measure the performance of a Neural Network. It is a measure of the total number of 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:

(1)

With the threshold for each of the models at 0.5, the accuracy is shown in Table 4. The accuracies of all the models are not very different so further performance metrics have been calculated.

**Table 4: Model Accuracy Results.**

| 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 |

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. Table 5 shows the sensitivity and specificity results. The equations for the calculation are as follows:

(2)

(3)

**Table 5: Sensitivity and Specificity Results.**

|  | 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 |

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. The higher sensitivity reveals a larger number of false positives, which then reduces the specificity. 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 BERT model and FNN model agree. In both, the sensitivity is very low. This tradeoff between the sensitivity and specificity can be seen in the plot in Figure 5. As the sensitivity increases, the specificity decreases.



**Figure 5: Sensitivity and Specificity Graph.**

## F1 Score

The F1 Score is another method of measuring the performance 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:

 ((4)

Using the classification rates, the following F1 scores were obtained and are shown in Table 6.

**Table 6: F1 Score Results.**

| 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 |

However, caution should be taken when using the F1 score for evaluating the model performance when there is a biased dataset, like the one used in these models, because it is independent of the number of true negatives [6]. Only 25% of the sentences in this dataset are requirements. The F1 score does not account for the correctly identified non-requirements in the sample, which makes up a large portion of the sentences.

## Matthews Correlation Coefficient

Mathews Correlation Coefficient (MCC) 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 [6]. The equation for calculating the MCC and the MCC for each model is shown below and in Table 7:

 ((5)

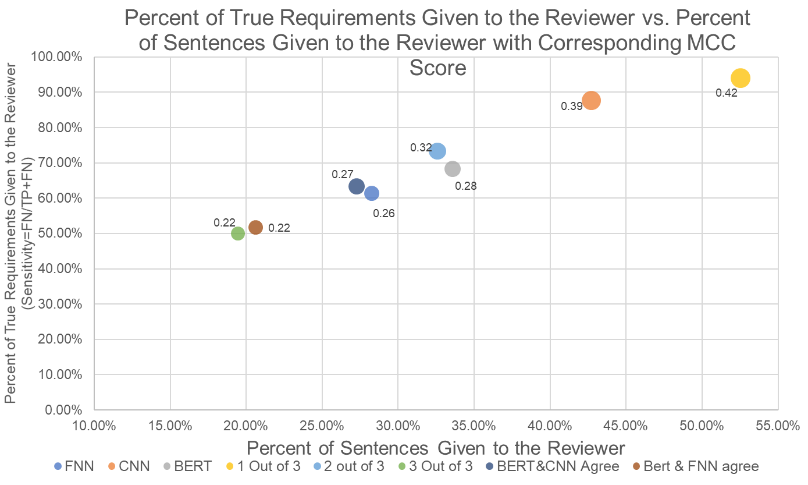
**Table 7: Matthews Correlation Coefficient Results.**

| 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 |

## Analysis of Strategy Alternatives

Reviewers of the requirements documents may have different preferences for how much they trust the AI tool. Because of that, different strategies have been developed to meet the objectives of different reviewers. If the objective of the reviewer is to reduce the most workload on reviewing a document and accept the reduction of true requirements identified, 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 identify the most true requirements, but accept a greater workload, then a set of predictions with a higher MCC and lower false negative rate should be used. This reveals a tradeoff between the total number of requirements to review (workload) and the total true requirements identified by the AI.

The strategies can be derived from the following plot in Figure 6. The x-axis shows the percent of sentences given to the reviewer and the y-axis shows the percent of true requirements given to the reviewer (sensitivity). Each data point in the plot is a different set of model predictions. Next to each plot is the corresponding calculated MCC. This plot reveals a linear pattern among the sets of predictions. As the percent of sentences given to the reviewer increases, the percent of true requirements given to the reviewer increases. This relationship allows the reviewer to choose which set of predictions they would want to use depending on their trust in the AI and their workload preferences.



**Figure 6: Percent of True Requirements Given to the Reviewer vs. Percent of Sentences Given to the Reviewer.**

Each of the strategies that a reviewer may select are described below.

### 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 and will receive 93% of the true requirements. In this case, however, there will be many false positives.

### 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 document would need to be reviewed.

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

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%.

An improvement in the models would show a decrease in the slope of this line and shift it up and to the left. This would increase 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.

# Impact

The results show that by using the AI, a reviewer can reduce review load to the systems engineer by 50% and still identify 90% of the true requirements. With another perspective, the reviewer can reduce the review load by 80% with the identification of 50% of the true requirements.

A final impact by this project is the identification of future paths to take AI for requirements identification and analysis and the identification of challenges with implementing AI for requirements identification. These are discussed in the conclusion.

# 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. One of MITRE’s tasks in this effort was the system must also be modular for future improvements. Future improvements on the system could be:

### 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 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

##### Acknowledgment

Luke Fritzke, Eric J. Newman, and Jenny Rekas, served as MITRE customer representatives during the duration of this project. Dr. Songjun Luo oversaw the project as the GMU SEOR faculty representative.

##### References

1. G. Lami, “QuARS: A Tool for Analyzing Requirements,” Presented at Software Engineering Measurement and Analysis Initiative. [Online]. Available: https://resources.sei.cmu.edu/asset\_files/TechnicalReport/2005\_005\_001\_14603.pdf
2. K. Verma and A. Kass, “Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents,” Accenture Technology Labs, San Jose, CA, USA, 2008.
3. Respect-IT. *A KAOS Tutorial.* (2007). Accessed: Feb. 2021. [Online]. Available: http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf
4. M. Petrotta and T. Peterson, “Implementing Augmented Intelligence in Systems Engineering,” *NDIA Ground Veh. Sys. Eng. And Tech. Symp.*, (Novi, MI), 2018. [Online]. Available: http://gvsets.ndia-mich.org/documents/SE/2018/Implementing%20Augmented%20Intelligence%20in%20Systems%20Engineering.pdf
5. *Natural Language Requirements Dataset*, National Research Council of Italy. [Online]. Available: http://nlreqdataset.isti.cnr.it/
6. 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, Jan. 2020. Accessed: Feb. 2021. [Online]. Available: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-019-6413-7
7. *Bidiriectional Encoder Representations from Transformers (BERT)*, TensorFlow Hub. [Online]. Available: https://tfhub.dev/tensorflow/bert\_en\_uncased\_L-12\_H-768\_A-12/2
8. J. Brownlee, Deep Learning for Natural Language Processing.