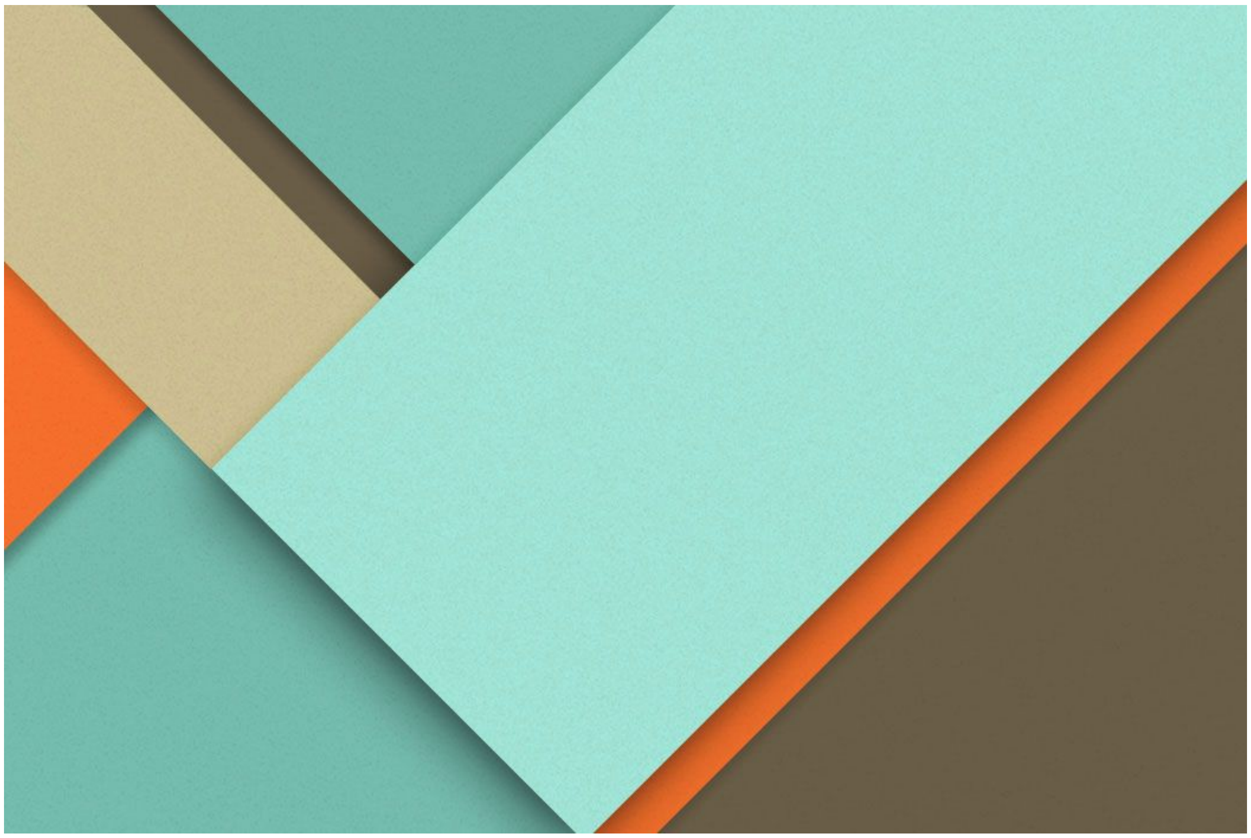


Artificially Intelligent Requirement Analysis Tool

Design Document Team 8

03.26.2019

SE 491

Team 8

1 Introduction

1.1 Acknowledgement

We would like to give special thanks to our faculty advisor, Dr. Mitra for helping us with planning this project. Also, to Jason Wong and the Collins Aerospace advisors for hosting and overseeing this project. Finally, a thanks to teams at Google who put together the Google Word2Vec Model which will give us a great beginning with a great NPL model.

1.2 Problem and Project Statement

Requirement tracing, which is the process of creating logical links between individual requirements, is essential in projects carried out by Collins Aerospace. When working with safety critical systems, it must be ensured that all necessary features are recognized, that no unnecessary features are included, and that we can link the reasoning for including a component to a higher feature. Projects at Collins Aerospace may include thousands of requirements, most of which link to one or more other requirements. As of today, employees at Collins create and review these requirements by hand or with "non-intelligent" tools. Manually reviewing the accuracy of a requirement trace and deciding which links are good and which need to be removed is extremely expensive in terms of the time that must be dedicated to ensure that the requirement trace is sufficient for the project at hand. The purpose of this Capstone project is to develop a tool for Collins in order to automate requirement trace analysis.

Our proposed solution to this problem is to utilize the Gensim python library which include algorithms such as Word2Vec and Word Movers Distance. These algorithms analyze the similarities between text. Our tool will take multiple Excel documents as input, which specify all individual requirements and the other requirements that each are linked to. By using the gensim libraries, we will be able to generate similarities between the linked requirements and feed them into a model which will then flag the link as either good, bad, or suspicious. The tool will also recommend possible links for requirements which may have had a bad link or if it finds a very likely match. Finally, a file containing a report generated by the analysis will be displayed for the user to view.

1.3 Operational Environment

This project is completely software based and therefore will operate on Collins Aerospace servers. Users will interact with a web GUI which communicates with our back end server which will be written in python. The major concerns associated with this project are security related, so we must ensure that all data given to our tool and output by our tool is secure.

1.4 Intended Users and Uses

Intended Users:

The users of this tool will be Collins Aerospace engineers. There will be several engineering teams that will have access to the tool, however they will all have the same intended uses which allows us to group them into one type of user.

Intended Uses:

There is only one major intended use for this tool. The tool will be used to generate reports on an analysis of requirement links. The process of analyzing these links will initially include classifying them as good, bad, or suspicious. Secondary functionality includes suggesting possible missing links between requirements. Once the analysis process completes, a report will be generated for the user to view and fix any errors found by the tool.

1.5 Assumptions and Limitations

Assumptions

- A web form will be sufficient for configuring settings in the analysis process.
- HTML, CSS, Bootstrap and JavaScript will be sufficient to develop the front-end UI.
- Similarity given by Word2Vec and Word Movers Distance will be sufficient to create a classification model for analyzing links.
- Google's Word2Vec model will be a more than sufficient model for our purposes, and will reduce time accounting for out of vocabulary words.
- System will be sufficient enough to handle multiple numerous user request

Limitations

Collins Aerospace has not put any limitations on this project. The project is open for our team to make these decisions.

1.6 Expected End Product and Deliverables

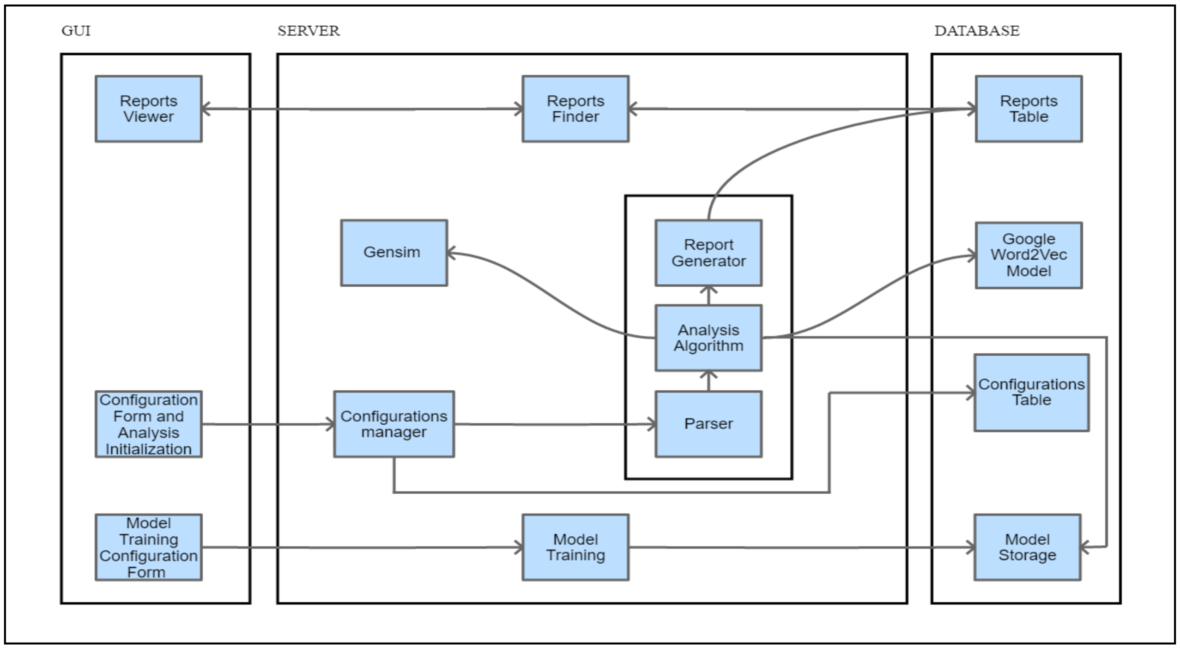
Our team plans to deliver a tool to read requirement documents provided to it by a user, analyze the documents, and classify links between requirements as either good, bad, or suspicious. The tool will be hosted on a linux server and will be capable of being accessed by either command line or web GUI. Secondary deliverables include expanding the tool to include the following features:

- Identify/suggest possible missing links between requirements
- Analyze links between code and requirements
- Analyze links between tests and code

2. Design Specifications

The Design of the software will contain three main components GUI, server and the database. The GUI will contain a report viewer, configuration form and analysis initialization, and model training configuration form. Server will contain components like reports finder, gensim libraries, configurations manager, parser, analysis algorithm, report generator and model training. The database will contain report table, google word2vec model, configuration table and model storage. Report functionality will take place based on report viewer, report finder and reports tables which will be generated by the server using other components. Other analysis algorithms will be based on user's selection of algorithm for analysing the link which will be obtained through the model training configuration form.

2.1 Proposed Design



2.1.1 Reports Viewer and Reports Finder:

The reports generated by this tool will be stored in the Reports Table in the database. These reports will be able to be viewed by the user via the Reports Viewer on the GUI. The user will go to the Reports Viewer and select a report they would like to view. The desired form will be sent to the Reports Finder module which will pull the report from the table and send the data back to be displayed in the Reports Viewer on the GUI. The user will also have the option to download the report into an excel file from this page.

2.1.2 Configuration Form/Analysis Initializer and the Configuration Manager:

The tool will need information from the user in order to run the analysis process. The information will be provided by a webform on the GUI. This information will include the following:

- The input requirement documents will be selected to be upload to the server for analysis
- Information about which requirement file links to which other requirement file
    - This may be also be implemented by enforcing a file format which declares a column only if the files are linked
- The input document layout (i.e. the column names) so that our parser knows how to read the input files
    - For simplicity, the file layout may become strict requiring the user to follow it
- Whether to scan for possible missing links
    - Not scanning for these missing links will make the process run much faster because there will be a reduced number of comparisons between requirements
- Whether to display or download the report when analysis is complete
    - If not specified to do either of these options, the report data will still be stored in the database.

Once the user has specified all of the information that the tool requires to run, the user will have three options for proceeding. First, they may start the analysis process with the settings provided in the form they filled out. Otherwise they could simply save the settings without running the tool, or finally, they could both run the tool and save the configurations at the same time.

The user also will be provided with the alternative option of providing and submitting a settings file. The parser will be set to read either option so that we may also run the tool via command line effectively. The analysis module will just need to be called with the respective settings file and the process can then be called or even automated or scheduled.

2.1.3 Parser

The parser is the first component of the analysis process. Before our tool can analyze any requirements, the parser will extract the data from the requirements documents provided by the user. The parser will receive parsing rules from either a GUI web form or a settings file specified by the user. From there, the parser will accomplish the following tasks:

- Check that the files contain the required columns which include:
    - Object ID
    - Requirement text
    - Derived
    - Object ID of linked requirement
- Replace any abbreviations with their full wording
- Extract the fields listed above into a dictionary which will be used by the analysis algorithm

2.1.4 Google's Word2Vec Model

In order to capture the similarity between requirement text, we will need to use a method for natural language processing. We chose to use Word2Vec for word embeddings. Due to the security of Collins Aerospace, we are unable to access enough data to efficiently train our own Word2Vec model. Luckly, Google trained a Word2Vec model which is free to the public. This model was trained on data from Google News and contains over a billion word embeddings. This will be a highly trained model which will allow us to spend our time away from training Word2Vec models and finding methods to account for out of vocabulary words.

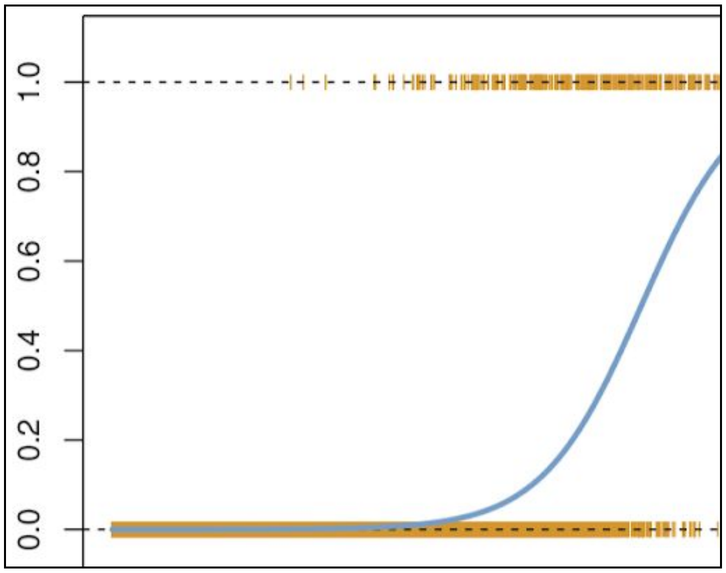
2.1.5 Gensim (Word Mover's Distance)

Word2Vec on its own, is not enough to calculate similarity between two pieces of text, or in our case, two requirements. This is where Gensims Word Mover's Distance algorithm comes into play. The Word Mover's Distance algorithm will take the two linked requirements as parameters and output a distance which represents the similarity between the requirements. The output is technically a distance where smaller distances declare high similarity between requirements and large distances represent low similarity.

2.1.6 Classification Model Training

From the Model Training Configuration Form, the user will specify requirement documents to upload for training a new model. These documents will be similar to real documents that are used in analysis, except they will have an extra column which specifies whether a link is good or bad. This column paired with output for the distance between requirements (similarity) will be what the model uses to learn.

Without knowing what the output from Google's Word2Vec and Gensim Word Mover's Distance will look like, we must predict how our data will look until we can actually visualize the true data. In theory, it should look something like the figure below:



The true classification of a link should either be good or bad. In the graph above, the y axis represents the class probability and the x axis represents the distance output by Word Mover's Distance (aka the how similar the links are). These classifications can be seen in the figure above where the yellow marks where Y = 0 are the good links plotted at their Word Mover's Distance, and the marks on the top are bad links plotted at Y = 0. The blue line is a sigmoidal function that fits the probability that any given point is either good or bad.

Say we wanted to classify a point using the graph above. We would plug in our Word Mover's Distance and see what the Y value is. Let's say Y = .4 at distance D. We would see that Y < .5 and say that the probability leans towards the link being a good link. Now if Y = .8, we would predict that the link was bad, because the probability leans towards the link being a bad link. What we can proceed to do is define a range where we classify the link as suspicious. To do this we may say that if .4 < Y < .6, then we cannot accurately predict the links class because it's nearly a 50% chance that the link is either good or bad. By following this format, we can read a distance provided by Word Mover's Distance and return a classification as good, bad or suspicious. To store the model in the database, we may simply store the coefficients of the sigmoidal function.

2.1.7 Analysis Algorithm

The analysis algorithm will bring together everything discussed so far. The algorithm will be given the dictionary of requirement data generated by the parser, will use Google's Word2Vec model with Gensim Word Mover's Distance algorithm, and will query for the desired classification model. For each link, the algorithm will use the Word Mover's Distance algorithm to evaluate the similarity of the requirements. The output will then be run through the classification model which determines if the link is good, bad, or suspicious. Each of the classifications will be stored and sent to the report generator once analysis is finished.

After primary functionality is implemented, we will begin designing and expanding this algorithm to suggest possible missing links, analyze links from code to requirements, and analyze links from tests to code.

2.1.8 Report Generation

When the analysis algorithm is finished, it will send all of its classification decisions to this report generation module. This module will store all the findings of the analysis into the Reports Table. If specified by the user, the user will be sent to the Reports Viewer page to review the report that has been generated.

2.1.9 Database

The structure of the database will include three tables: Settings, Reports, and Models. Whenever a user trains a new model, the coefficients of the models will be stored in the Models table, so that they may be accessed again later. If the user specifies that they would like to save their settings, the settings will be stored in the settings table. After an analysis has been completed, the report that is generated will be stored in the Reports table so that all past reports that have been generated can be accessed and viewed by the appropriate users.

2.1.10 User Interface

The tool will include a simple and clean web GUI that allows users to configure the tool to their needs. There will be four pages for the user to access which are as follows:

1. Reports Viewer
2. Configurations Form/Analysis Initiation
3. Model Training
4. About/Help Page

The functionality of the first three pages have been discussed above. The fourth page will be a reference for users when there are any questions or help needed. We will provide instructions on using the tool, and explanations of how process work so that the user will be able to use the tool to its full potential.

2.2 Current Status

2.2.1 Research Tools for NLP Word Embeddings

Our team started out the semester by researching different methods of calculating similarity between two pieces of text. We researched the following methods for embedding text into word vectors: Word2Vec, FastText, GLoVe, and Doc2Vec. During our research, we noticed that most of these methods were very similar in terms of input and output. With this being the case, we decided on Word2Vec. We did this because while researching Word2Vec, we found that Google released a Word2Vec model that was trained on Google News and contains over a billion word vectors. This will negate the problem we've had with finding enough data to train a sufficient Word2Vec model. With Google's Word2Vec model, we will also have less worries about encountering out of vocabulary words.

2.2.2 Observed Requirement Data

Once our team was eventually granted access to a small portion of requirement data at the Collins Research, we have been familiarizing ourselves with the requirement documents we are expected to be working with. These documents turned out to be much more complex than we expected. With this being the case, we are designing a method to parse the data and analyze requirements in a way that the Word Mover's Distance algorithm will be able to give reasonable results.

2.2.3 Conclusion of Work Done so Far

As mentioned above, Word2Vec and Word Mover Distance algorithm will be implemented as per the requirement and the data observed, which was provided to our team through Collins Aerospace. Till time, we had difficulty in searching for sufficient data for the model to be trained. We have requested Collins Aerospace to provide us with enough data to train the model, in order to provide them with sufficient and efficient classification of the links. Thus, as a solution to strengthen our weaknesses of insufficient data, we have decided to train our model with Google News.

2.2.4 Implementation Issues and Challenges

The project is related to the United States Defence Department sponsored by Collins Aerospace. So, far the algorithm has been implemented such as word2vec and doc2vec. However, to train the model based on these algorithms, we need sufficient data. Due to the security and classified information involved with the project, Collins Aerospace is having tough time sharing the required information to train the model. However, we have started training the model based on the minimum data provided. Other than that, we are asked to create dummy data for the training purposes. That been said, it might not be the most accurate trained model as the data it’s trained on is dummy and not the original one.

3 Testing and Implementation

3.1 Interface Specifications

The user will interact directly with one of two components to run the tool; either a command line, or a graphical user interface. The components that the user have access to will call the components on the server via an API that will be defined for communication between client and server. From there the server will communicate with the database via the standard MySQL interfacing.

3.2 Hardware and software

We will be using python’s built-in module “unittest” for testing parser, trained mode, report generator, or any other modules written in python. We will use unittest for testing the databases as well.

A test program called Postman will be used to manually test the REST API we will develop for the UI. For automated testing, we will simply use python's "requests" module and "unittest".

For UI, we will use mocha and selenium. Mocha is for unit-testing the functions used inside the UI. Selenium is used to test the UI such as simulating the click, input data, scroll etc.

Hardware - we will not use any hardware for our project.

3.3 Functional Testing

3.3.1 Testing Model Accuracy

The most important testing that we will need to do is testing how accurately our model can classify links. We would like to shoot for a minimum of 70%-80% classification accuracy. If our model has accuracy under this range, then we will need to either rethink our classification model, or come up with a requirements formatting standard that will help us capture similarity more accurately.

3.3.2 Testing Algorithm Under Configurations

Another major area for functional testing will be testing whether the analysis process will break under any combination of settings. The tool will be able to be configured in many different ways so we need to make sure that the tool won't break down and handles any poor configuration attempts.

3.3.3 Testing Parser with Broken Input

We will need to test our parser and make sure that it will not break if the tool was misconfigured or the input data is improperly formatted. It is very possible that these situations occur due to user error or updates in the company processes. We need to test the parser to ensure that it will be able to respond appropriately under all circumstances.

3.3.4 Testing Methods

Our team decided that using the standard python testing library unittest will be sufficient to test all of our projects components. The library is both well received and trusted so we will have no conflicts in our decision of testing library.

One component will be tested at a time. Once a two communicating components are individually tested, they will be put together and tested as a whole to make sure they work as expected. This process will continue until all the components have been tested and implemented together.

3.4 Non-Functional Testing

3.4.1 Security

Security is important for this tool as the data deals with information provided by U.S Defence Division through Collins Aerospace. Following are the testing that needs to be performed before the end product is handed over to Collins Aerospace:

- Vulnerability Scanning

  The software needs to be scanned for any vulnerability that may cause any security holes.

- Security Scanning

  Perform testing to identify weakness related to network and system and reduce the risk of such security holes using manual or automated system that are already available.

- Penetration Scanning

  Testing will be performed on the software to evaluate if there are any changes of malicious hacker to hack the system easily. In order to reduce the risk of such attacks, analyses needs to be performed on the system related to security.

- Posture Assessment

  Security scanning related to ethical hacking and risk assessment needs to be performed in order to verify that there are no risk related to the software that could cause loss of data as the data is highly classified.

3.4.2 Portability

The application will be used in different department of Collins Aerospace. The software needs to be deployed easily and should be able to perform its duty as per each department requirement. Thus, the portability of the software comes in handy and needs to be flexible for each department.

3.4.3.Performance

Application will be forming machine learning algorithm on large number of data provided by each department of Collins Aerospace. Thus, it needs to be done as effectively as possible so that the processes are not time consuming and inefficient. Thus, performance is another important factor that needs to be initiated while designing the software.

3.4.4 Modifiability

As the application is going to be used by different department of collins aerospace. Each department has an option to implement their own machine learning algorithm. Thus, it needs be easily modifiable so that no other section of the software breaks on a new updated version of the software.

3.5 Process

Section 1: Training Module

- Test the accuracy of trained model using test data (we will have both train data and test data).
- Statistically show how accurate it is (e.g. xx %). Since there is a trade-off between train data accuracy and test data accuracy, choose the configuration that creates the max value of the sum of both accuracies.

Section 2: Analysis Algorithm

- Core of our application and the most important. We need to thoroughly test this module. Again, we will use python unit test.
- There are two important aspects: Accuracy and Efficiency. The accuracy can be tested by randomly choosing input with known results and making a prediction. Then check how accurate the prediction is.
- The efficiency can be tested by using the timer. Ideally this should be in a matter of seconds since developers need to use our tool very often.
- Now check to see if the algorithm raises any exceptions. The most common exception would be out-of-vocabrary exceptions. We will check this will not occur for all the data we are provided so far.

Section 3: Parser

- Given data with enough edge cases, check if the parser gives the output with desired format.
- Check if all acronyms are replaced.
- Check if all flag type codes are replaced with English texts that hold some meaning.
- We will use python unit test.

Section 4: Web User Interface

- Both manual testing and automated testing will be performed. Test for both Firefox and Chrome.
- Open a webpage with a browser
- For reports viewer, with given report key in the input box, find the corresponding report
- if the report shows up, then check if the report has all the information we need
- For configuration UI, check if the UI have all necessary inputs
- Then check if the inputs a user provides are actually sent to the server by checking the server log.
- If the input is sent to the server, check if the data is in the correct format.
- (Optional) From the security perspective, check if all forms on the UI are resilient to XSS and Infection attacks. This can be done by using Kali and the associated tools.

Section 5: Configuration Manager

- First, We will check if it contains all necessary configuration options.
- Then check if the configurations are properly stored in the database.
- It needs to endure the concurrency problems since multiple users can edit the configuration at the same time.

Section 6: Report Module

- This is an intermediate step from analysis algorithm output to the database.
- Unit test if the data is properly received
- Unit test if the output is stored properly in the database by using raw SQL *statements.*

Section 7: The whole report generating process

- *This process has to be able to handle multiple requests at the same time. So input multiple data and see if it doesn't stop working.*
- *Anticipate the hardware failure. Occasionally store the intermediate results into the disk to mitigate hardware failure. During the testing, we will force shutdown the system and see if it recovers.*

3.6 Results

At this point in our project, we do not have any finished prototypes and therefore have not been able to perform any tests of our implementation. Therefore, we are unable to make any statements about results that have been obtained at this time. Once we have reached a point when we have performed tests, this document will be updated with results.

4 Closing Material

4.1 Conclusion

Artificial Intelligence for Requirement Analysis Tools will be delivered successfully along with a GUI that contains a command line application as well as other visual and form-based application as asked by the client. The software will be focused on analysis of links between the platforms, domains and subdomains as per the client’s requirement. Analyzing will be performed using machine learning algorithms such as word2vec, doc2vec and Word Mover Distance. The software will meet the non functional requirements related to security, performance, portability, and modifiability. These project will help the company to analyze the system links, and would reduce human errors.

So far, our team have been researching, implementing and testing word2vec, doc2vec and word mover distance algorithm. Apart from that, a GUI is under progress, and would be completed soon before the end of the semester. However, the testing of the GUI will be performed as we go through the process as well as in the upcoming semester.

Future goals includes implementation of a parser, an algorithm for classification of the links, visual representation of data on the GUI, and testing the software based on functional and non-functional requirements. At the end of the process, our team will be delivering the end product to the client along with the documentation describing the process to use the software as well a brief video tutorial on the software implementation and usability.

4.2 References

“gensim: topic modelling for humans,” *Radim ÅehÅ¯Åek: Machine learning consulting*. [Online]. Available: https://radimrehurek.com/gensim/models/word2vec.html. [Accessed: 26-Mar-2019].

“gensim: topic modelling for humans,” *Radim ÅehÅ¯Åek: Machine learning consulting*. [Online]. Available: https://radimrehurek.com/gensim/models/doc2vec.html. [Accessed: 26-Mar-2019].

G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning: with applications in R*. New York: Springer, 2017.