



Collins Aerospace Artificially Intelligent Requirement Analysis Tool

Project Plan Team 8

04.24.2019

–

SE 491

Team 8

Members:

Apurva Patel - *Project Lead, Report Manager, Technical Support, Communicator, AI Training Lead*

Ryan Cerveny - *Scrum Master, Meeting Scribe/Facilitator, Project Lead, Communicator*

Takao Shibamoto - *Chief Engineer, Researcher, UI Lead*

Jonathan Murphy - *Testing Engineer, Researcher, Requirement Lead*

Client:

Collins Aerospace (Representative: Jason Wong)


Faculty Advisor

Dr. Simanta Mitra

Table of Content

Project Plan Team 8	0
Table of Content	1
List of Figures	3
List of Table	4
List of Symbols	4
List of Definition & Acronyms	4
1. Introductory Material	5
1.1 Acknowledgement	5
1.2 Problem Statement	6
2. Project Deliverables and Specifications	6
2.1 Deliverables:	6
2.2 Project Specification	7
2.2.1 Operating environment	7
2.2.2 Intended Users and Use Cases	7
Figure 1: Use Case Diagram	8
2.2.3 Assumptions and Limitation	8
3. Previous Work and Literature Review	9
Figure 2: Autonomous Generation Code Generation Framework	10
4. Proposed Design/Solution	11
4.1 High Level Block Diagram	11
Figure 3: Block Diagram for AI-RAT System	12
4.1.1 Command Line Input	12
4.1.2 Requirement Directory Parser	12
4.1.3 Topic Modeling Module	12
4.1.4 Gensim Library for Topic Modeling	12
4.1.5 Link Classification Algorithm	12
4.1.6 Reports Module	13
4.1.7 Report Storage Directory	13
4.2 Functional Requirement	13
4.3 Constraints Consideration	13
4.3.1 Limited Access to Data	14

4.3.2 On Campus Accessibility of Restricted Data	14
4.4 Technology Consideration	14
4.5 Security Consideration	14
4.6 Safety Consideration	14
4.7 Technical Approach Consideration	14
4.7.1 General Approach	15
4.7.2 Word2Vec & Word Movers Distance	15
4.7.3 Doc2Vec	15
4.7.4 GloVe	16
4.7.4 Topic Modeling	17
4.8 Validation and Acceptance Test	18
4.9 Cost Consideration	19
4.10 Possible Risks and Risk Management	19
4.11 Feasibility Analysis	20
4.11.1 Unsupervised Learning:	20
4.11.2 Providing Topics/Probability Likelihoods:	20
4.11.3 Trainability:	20
4.11.4 Conclusion of Feasibility:	21
4.12 Project Proposed Milestones and Evaluation Criteria	21
4.13 Project Tracking Procedure	21
4.14 Test Plan	22
4.14.1 Forward Traceability Test	22
Figure 5: Test Case for Forward Traceability	22
4.14.2 Backward Traceability Test	22
Figure 6: Test Case for Backward Traceability	23
4.14.3 Bi-Directional Traceability Test	23
Figure 7: Test Case for Bi-Directional Traceability	23
4.14.4 Classification and Prediction Test	23
5. Assessment of Proposed Solution	24
5.1 Summary of suggested approaches	24
5.2 Assessment of Word2Vec and Word Mover's Distance	24
5.3 Assessment of Glove	24
5.4 Assessment of topic modeling	25
6. Estimated Resources and Project Timeline	25
6.1 Estimated Resources	25
6.1.1 Personnel effort requirement	25
Table 1: Personnel Effort Requirement	27
6.1.2 Other resource requirement	27



6.1.3 Financial requirement	27
6.2 Project Timeline	28
Figure 8: Tentative Gantt Chart	29
7. Standards	30
7.1 Unethical Processes	30
7.2 Team and Company Interactions	30
7.3 Security	30
8. Closure Material	31
8.1 Closing Summary	31
8.2 References	32

List of Figures

1. Figure 1: Use Case Diagram
2. Figure 2: Autonomous Software Code Generation
3. Figure 3: Block Diagram
4. Figure 4: Results from GloVE Algorithm
5. Figure 5: Test Case for Backward Traceability
6. Figure 7: Test Case for Bi-Directional Traceability
7. Figure 8: Tentative Gantt Chart

List of Table

1. Table 1: Personnel Effort Requirement

List of Symbols

1. Symbols 1: x_i is training
2. Symbols 2: L_e is a set of existing links
3. Symbols 3: L_p is a set of predicted links
4. Symbols 4: \cap intersection
5. Symbols 5: ξ_i is a slack variable that allows x_i to be misclassified if necessary

List of Definition & Acronyms

1. Requirement Tracing

The process of tracing or recording the links between the higher level requirement system with other individual lower or other higher level requirement system.

2. Requirement Gathering

The process of gathering the specification as the requirements such as use cases, higher level requirement, functional requirement, technical requirement and many more from the stakeholders that will be used as the formal data for requirement tracing.

3. Supervised learning

Machine learning task to related every input with a desired learned output based on the example output input pair.

4. Unsupervised learning

Machine learning task to draw inferences from dataset that includes inputs with no labeled response.

5. NLTK

The Natural Language Toolkit. A python library that was used for stop words.

6. Confusion Matrix

It is a table with rows and columns that reports the number of false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN).

7. LDA - Latent Dirichlet Allocation, one of the method for Topic Modelling

1. Introductory Material

1.1 Acknowledgement

We want to acknowledge all our clients at Collins Aerospace, specially to Jason Wong, Kathleen Knott, and Branden Lange for supporting us and providing us with feedbacks and resources for carrying out the project. We also want to acknowledge our faculty advisor Dr. Simanta Mitra for guiding through the development process, college of engineering and Dr. Thomas Daniels for providing us guidance, expertise and necessary hosting resources.

1.2 Problem 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 topic modeling functions of the Gensim Python library. These algorithms analyze the similarities between text by determining the topic of the text through natural language processing techniques that analyze the context, and comparing the determined topic to the topics of relating to other texts in order to calculate similarity. 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 stored for the user to view.

2. Project Deliverables and Specifications

2.1 Deliverables:

Collins Aerospace will be provided with the following deliverables:

- Well commented source code
- Executable for the Tool (batch file)
- Instruction Manual such as written documentation
- Step-by-Step video instruction manual on how to rebuild the executable and how to setup the tool
 - Updated READ_ME.txt file with a list of required tools and libraries such as follows:
 - Python 3.7
 - Anaconda 3
 - Gensim
 - Matplotlib
 - Pandas
 - Numpy
 - NLTK
 - Scikit learn
 - Spacy
 - Scipy
- Step-by-Step video on how to use the tool
 - Short Tutorials on different segments of the tools in order to help the engineer at Collins Aerospace to quickly grasp the tool
- Updated UML Diagram, Use Case Diagram, Block Diagram
- Documentation related to the algorithms like Word2Vec, Word Mover Distance, GloVE, Sentence Encoding, Topic Modeling. The following document will include a brief explanation to all the above algorithms along with their pros and cons related to the project. Moreover, it will also include a status related to the use of the algorithm based on the project's final result.
- Deliver a live presentation of the final product

2.2 Project Specification

2.2.1 Operating environment

Collins Aerospace Artificial Intelligence for Requirement Analysis Tool is a software based project. The tool will be operated on the Collins Aerospace server. The user will operate the tool using a command line. Any operating system that has a basic command line would work. The tool will require python3.7, gensim, scikit learn, scipy, spacy, matplotlib, nltk, anaconda 3 and Windows/Linux/iOS operating system to function as per the requirements. The following tool process a sensational data related

to security which needs the tool to be highly secure. Thus, secure environment along with a securely implemented tool is a must.

2.2.2 Intended Users and Use Cases

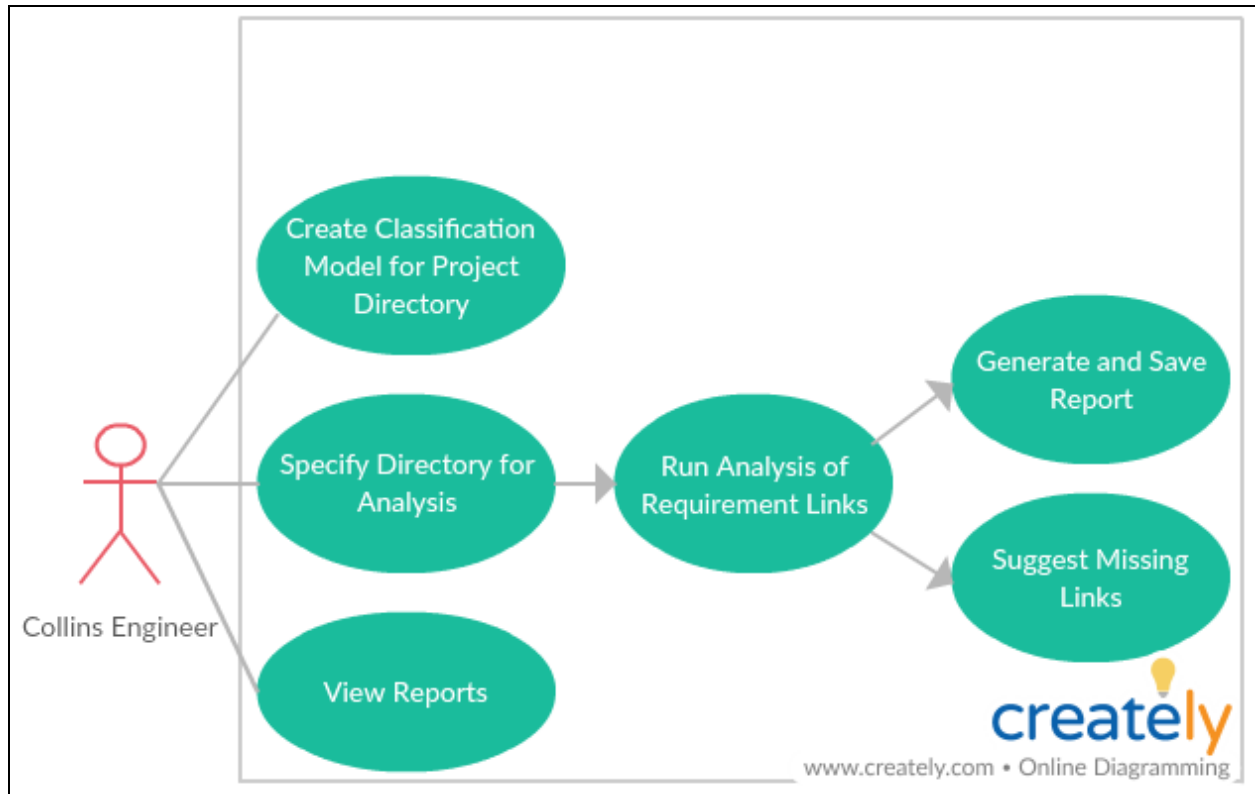


Figure 1: Use Case Diagram

Intended Users:

The engineers at different departments at Collins Aerospace will be using the tool. Each department will have their own tool configured separately. Thus, every engineer would access the tool as a common user.

Intended Uses:

Artificial Intelligence for requirement analysis tools is a software based tool that will be used as requirement tracing. The main uses of this tool is to classify the higher level requirement link and lower level requirement links as good, bad and suspicious. Moreover, it will also provide a prediction of lower level requirement links that relates to the higher level requirement. At the end of the analysis and prediction process, a report will be generated mentioning the good, bad, and suspicious link along with the prediction of the links.

2.2.3 Assumptions and Limitation

Assumptions:

- A configuration file mentioning the higher level platform and lower level platform and how they relate will be provided in the form of settings.py.
- The user will operate the command line when new data is been throw into the AI.
- A group of users will use the tool as a single type of user, any amount of groups can access the tools.
- Clean and readable output will be provided, representing the data with an appropriate color coding along with detailed labeling.

Limitations:

- The size of the data varies from department to department.
- The format of the input file will vary.
- The training model needs to be configured continuously in order to make the Artificial Intelligence tool continue to learn newly updated terms and definitions.

3. Previous Work and Literature Review

Requirement Tracing using Artificial Intelligence has been followed by various companies based on their requirement. Research has been conducted on algorithms to use for requirement tracing analysis. We referenced few of them. However, the closet one we referenced most of the time was “Use of Artificial Intelligence in Software Development Life Cycle”[1].

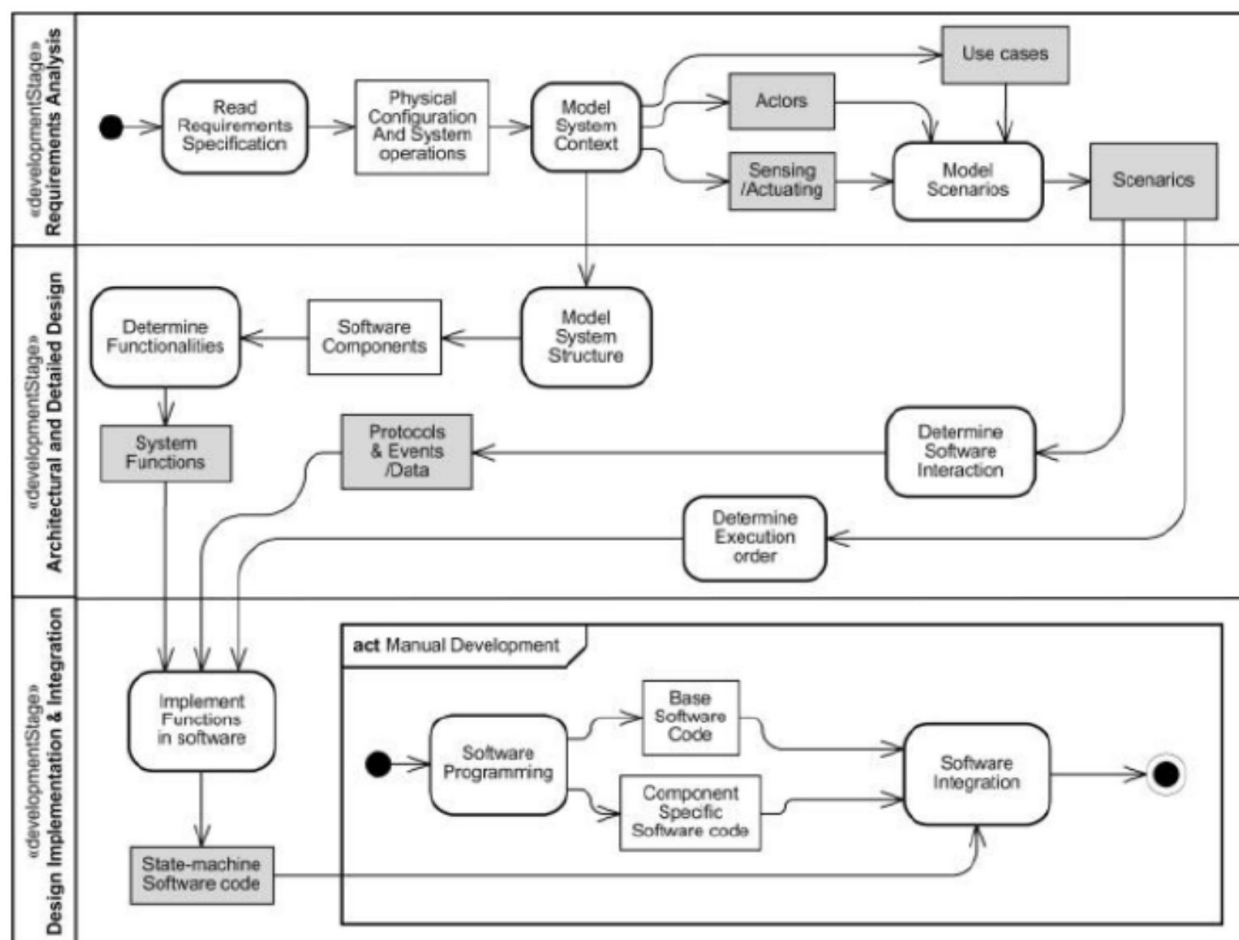


Figure 2: Autonomous Generation Code Generation Framework

The following figure is taken from the journal introduced in “Use of Artificial Intelligence in Software Development Life Cycle”[1]. In the following research, the system development stage is broken into three different stages: Requirements Analysis, Architectural and Detailed Design, and Design Implementation & Integration. In the Requirement Analysis stage, a requirement is been read, once the specifications are read, physical configuration and system operations takes place. After the specified operations are completed, a model is been trained based on the use cases, scenarios, and actors. The trained model is

passed onto Architectural and Detailed Design stage. In these stage, all the processed data is been taken as data. Moreover, software components are been introduced with system functionalities along with protocols and event data based on the previous scenarios obtained from the trained model. Once the Architectural and Detailed Design stage is completed, the system functions, protocols and event datas, and execution orders are transferred to the Design Implementation and Integration stage. In this stage, the function are implementation into the software, and a state machine software code is been generated which will be passed onto the software integration.

Another research that we looked into is from “Tracing Requirements as a problem of machine learning”[2]. The following research is based on the traceability approach. It explains the relation between the higher level requirement and the lower requirement. The classifier of the link is broken into parts, such as, the classifier is trained based on the four folds and is been tested and evaluated based on the remaining one. Moreover, every instance is represented as similar words, different words, verb pairs, verb group pairs, noun pairs, noun group pairs, and dependency pairs. Later in the research documentation, to understand the rationale framework, they introduces a formula as follows:

$$\frac{1}{2} |w|^2 + C \sum_i \xi_i \quad \forall i : C_i w \cdot x_i \geq 1 - \xi_i, \xi_i > 0$$

x_i is training, $C_i \in \{-1, 1\}$ is the class label of x_i ;

ξ_i is a slack variable that allows x_i to be misclassified if necessary, and $C > 0$ is the misclassification penalty”[2]

Based on the above classification, the researchers worked on the traceability of the requirements. Few of the approach was supervised learning and some was based on the unsupervised learning.

The artificially intelligent requirement analysis tool is a completely new project for Collins Aerospace, therefore we began this project from scratch and did not have any previous work to build our system off of. We reviewed documentation from similar projects that have been implemented in the past, and found a variety of other natural language processing tools that had been developed in order to take an artificially intelligent approach toward requirement analysis.

Our project reflects somewhat of the above descriptions. However, it has a different approach. The dataset provided to us contains manually created acronyms and flags by the Collins Aerospace engineers. Thus, these acronyms and flags are difficult to consider as a “natural language” wordings. Such acronyms and flags needs to be learned by the training model in order to classify and predict with high accuracy. For the same purpose, we will be using a bit of supervised as well as unsupervised learning algorithms. Topic modellings and sentence encoding similarities algorithms will be implemented in classifying the links as good, bad and suspicious. Moreover, these algorithms will be used to predict the links related to the higher level requirement.

4. Proposed Design/Solution

4.1 High Level Block Diagram

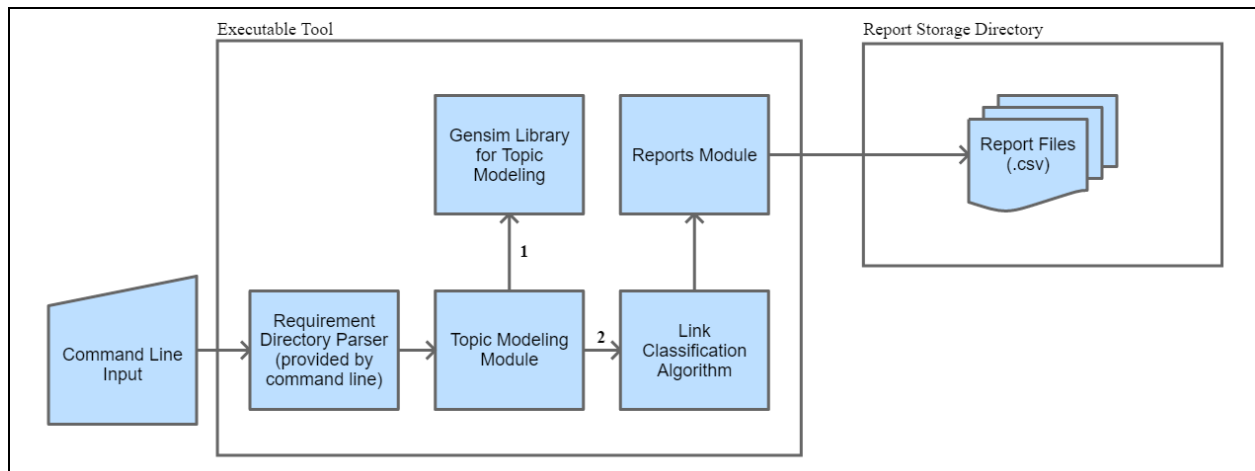


Figure 3: Block Diagram for AI-RAT System

4.1.1 Command Line Input

The tool will be launched via a .bat file. Therefore, the user will interact with the tool via the command line. Having the tool accessible via the command line is beneficial because it allows for easy automation/scheduling for analysis of various projects. The user will run our executable with the requirements directory they would like to analyze as a parameter.

4.1.2 Requirement Directory Parser

Once the tool has been called with the requirement directory as a parameter, the designated directory must be parsed and cleaned before topic modeling occurs. Since not each requirement directory is formatted in the same way, the directory will contain a file containing parsing rules for the directory. Once requirements are extracted, they will be cleaned and passed to the Topic Modeling Module.

4.1.3 Topic Modeling Module

The Topic Modeling Module is fairly straight forward. The module will receive clean requirement data from the parser, then proceed with a method call to the Gensim library. The Topic Modeling Module will call the Gensim LDA topic modeling method to return a topic model for the given directory. Once our topic model is created, the model is passed to the Link Classification Algorithm to be used for analyzing the requirements in the project.

4.1.4 Gensim Library for Topic Modeling

The Gensim python library is written to aid machine learning projects. From this library, we will be using the topic modeling API to train our model. The API can be found in reference [8].

4.1.5 Link Classification Algorithm

The Link Classification Algorithm will be made simple by the topic modeling approach. The algorithm will iterate over each link of requirements and use the previously generated topic model to produce a likelihood that the given link is good. Since the quality of the link is provided as a likelihood, we can create thresholds to create additional classifications such as suspicious links and bad links. For an example, links with a likelihood of $> 70\%$ would be classified as good links while a likelihood in the range (50%-70%) would be marked as suspicious. Finally, any likelihood of below 50% would be considered as a bad link. Along with this statistic for each link, the topic modeling approach will make suggesting similar links very easy, as it can list the x most similar links by likelihood. This data will all be sent to the reports module when the analysis process is finished.

4.1.6 Reports Module

The Reports Module wraps up the process by generating a report of the tools findings. The reports will be stored as a csv file inside the Report Storage Directory where the users will have access to them. Reports will contain classifications of every link from the input directory, along with the link's respective object IDs, requirement texts, and the likelihood that was used to make the links classification. Secondly, the reports will contain suggestions for links should there be a strong likelihood between requirements that are not linked already. Should it be desired, information about the model created to analyze the requirements may be stored here as well.

4.1.7 Report Storage Directory

When a new project directory is analyzed with this tool, a new directory will be created within the Report Storage Directory to store that project's reports. The next time that the project directory is analyzed, the reports will be versioned and stored in the same directory, so that a history may be stored for runs of each project.

4.2 Functional Requirement

- Tool should be able to identify the links as good, bad and suspicious
- Tool should be able to predict the related links to higher level requirement links
- Tool should be easy to configure between the teams
- Tool must be executable using command line
- Tool must be easily deployable on new machines
- The tool is able to handle or negate out of vocabulary words

4.3 Constraints Consideration

Collins Aerospace are dedicated clients to the defense of the United States of America. For the same reason, they must create data constraints to a student managed project outside the campus of the Collins Aerospace. Following are the limitation that needs to be considered:

4.3.1 Limited Access to Data

Being a defense oriented organization, the information contained in Collins Aerospace's requirement datasets is classified and access is strictly limited to employees of Collins. Thus, in an outsourced student project, it is difficult to share data with students due to these limitations. Sharing such data could result in a security issue, not just for the company, but for the United States as well. Thus, we have only been provided with a small fraction of data that has been approved by the Collins Aerospace.

4.3.2 On Campus Accessibility of Restricted Data

As the requirement data shared with us by the Collins Aerospace is classified, we are only allowed to access their data when we are present in their research facility. Thus, we have been full time students and part-time employees. When we need to test the software, we must make appointments when all the team members are available to visit the Collins Aerospace research facility in the Iowa State research park. Thus, data accessibility has become a time constraint.

4.4 Technology Consideration

Effectively training a model will require feeding it a large quantity of training data. Processing this information and using it to train the model will require high processing power which not all technology will have. We will have to keep this in consideration to make sure we have efficient platforms to train our models in a reasonable amount of time.

4.5 Security Consideration

This tool will be used by Collins Aerospace, which is a government facility that advances the country's military and aerospace technology. The input to the tool will be requirements for their projects which should not be exposed in any way to the public or other companies. Keeping this data secure will be critical for our project.

4.6 Safety Consideration

Since this project is entirely software based, we will not be working with any components other than our computers, therefore there are no safety considerations that need to be taken into account. There is also no safety considerations for the user, since the end product will be a software tool.

4.7 Technical Approach Consideration

Most of the experiments we have done so far are unsupervised learning. We have considered supervised learning as well since that would solve the problem that unsupervised learning could become a black box that we don't exactly know how it works. However, the problem with supervised learning for our project is that unless the program knows what the project is exactly about the program won't be able to effectively use the labels (the file that contains links)

4.7.1 General Approach

Data analysis is an important aspect of the project. In order to perform the related tasks we decided to have a general approach that includes data cleaning, exploratory data analysis and lastly the prediction [9].

For data cleaning phase, we lowercase all the words and remove all punctuations. We also found that only taking nouns in the REQ/UC effectively filters out unimportant words. For example, "system shall implement a bcc capability on the linux mail server" would be cleaned to "system bcc linux mail server". It's obvious that only keywords are taken and this would make the prediction much more accurate.

For exploratory data analysis, we made a wordcloud of most frequently used words. This step makes it easy to figure out what words that should be regarded as stop words.


For the last step we have experimented with various methods such as word2vec/word mover's distance, doc2vec, glove, and topic modeling.

4.7.2 Word2Vec & Word Movers Distance

Word2Vec is a word embedding software which we used via the Gensim python library. In short, textual data is first provided to a Word2Vec model. From there, the model creates unique vectors which store the "meaning" or context of each provided word. Words of similar meaning or context will be located in the same general vector space. More in depth explanation and visualization can be found in reference [6].

Once the Word2Vec model is trained, the Word Mover's Distance algorithm comes into play to calculate similarity between texts, or in our case, requirements. Word Mover's Distance uses the Word2Vec vectors from the first text, and then finds the minimum distance to translate to the Word2Vec vectors in the second text. The smaller the distance it takes Word Mover's Distance to translate between the sentences, the more similar they are. More in depth explanation and visualization can be found in reference [7].

The idea behind using Word2Vec and Word Mover's Distance is that requirements that should be linked, should have text that are similar. This approach failed us for two major reasons. First off, Collins requirement data contains several acronyms and flags. Whenever a new acronym or flag is seen, the model would need to be retrained to account for the new words. The problem is that there must be



extensive data to accurately capture the context/meaning of the flag, which was not available. Secondly, requirement data at Collins is not written in such a way that the algorithm will succeed. The algorithm thrives with sentences which are rephrasings of each other, but requirement texts varied too much for the algorithm to pick up similarity regardless of whether the links were classified as good or bad.

4.7.3 Doc2Vec

Doc2vec is based on word2vec, but the difference is that it considers the word order into account. With the dependency of Word2Vec, the result was similar to Word2Vec and Word Mover's Distance. So we decided to move on with different approaches. The lessons learned is that word orders do not matter in requirements tracing.

4.7.4 GloVe

GloVe, short for Global Vectors, is a common approach for determining the semantic similarity of words. This comparison is done through embedding the words to be compared, then calculating the cosine similarity between them (e.g. the cosine of the angle between two vectors of a product space that can be used to measure the similarity between the vectors) based on their positions in the multi-dimensional global vectors. The cosine similarity is returned as a decimal number representing the similarity of the words, where two of the same words are given the maximum similarity value of 1. For example, when comparing the words "man" and "woman", the resulting cosine similarity would be very large, but very small if you were comparing two completely unrelated words. This similarity comparison approach is very similar to Word2Vec (which is discussed in section 4.7.2). Though GloVe is very effective in calculating the similarity between two words, it is not nearly as effective in computing semantic similarity of sentences due to the fact that it computes similarity based solely on the similarity between words in a sentence but does not take into consideration the context in which those words are being used.

Studies comparing Word2Vec and GloVe ability to perform sentence comparisons have been conducted by Yves Piersman[5], and have concluded that Word2Vec is actually the better option when comparing sentence similarity, even though it is still not very good. The results showed that the average similarity of good links was very close to the average similarity of bad links, which would not allow us to effectively distinguish whether a link was good or bad in order to officially classify it. A screenshot of the results from an experiment we ran on a small set of requirement data can be seen below, displaying that the average similarity for good links (the total sum of the similarity / the total number of good links) and the average similarity of bad links only vary by 0.02418, making the two classifications indistinguishable from each other. After this experiment failed, we decided that we needed to take a different approach towards natural language processing, and try to find a method to compare sentences based on the topic of the content rather than strictly on the similarity of the text.

GOOD AVG: 0.7640965392517398
 BAD AVG: 0.7399165588195102

Figure 4: Results from GloVe experiment

4.7.4 Topic Modeling

The idea of topic modeling came up in our discussion after the experiments that showed that word2vec, WMD, and Glove which are the approaches that directly compare the similarity of texts turned out to make really bad predictions. Topic modeling approach, on the other hand, doesn't directly compare the similarity of two texts. Instead, it compares the topic probability distribution of two texts. So far it is showing some interesting results and we are focusing on this approach.

We have used the most popular topic model called Latent Dirichlet allocation (LDA). It puts that each document (in our case, each REQ/UC) is a mixture of a small number of topics with weight and each topic constitutes of small number of related words with weight. [4]

There are multiple factors to consider in topic modeling. There are mainly two things we need to consider: 1) how to make topics and 2) how to compare topics. We performed experiments with slightly different variations and the following is what we have tried.

Topic modeling experiment 1 - Separate LDA models for REQs and UCs

1. Find top N most used words and prompt engineers to choose stop words (this technique would apply to any kind of algorithms we would use)
 - eg. we probably don't need "shall" in REQ and "user" in UC
 - maybe completely remove any verbs
2. Somehow unify similar words
 - eg. unify "misspell", "misspells", "misspelling", "misspelled" into "misspell"
3. Make separate LDA models for REQs and UCS
4. Compare the probability distribution of words for each topic for each req for each uc
5. Each REQ chooses top X similar UCs

Topic Modeling Experiment 2 - One LDA model for both REQ/UC

1. Filter out all words except nouns
2. Combine REQs and UCs into one table
3. Make a LDA model with N topics
4. Choose the topic which has the highest score for each REQ/UC
5. For each REQ, find UCs that are in the same topic

Topic Modeling Experiment 3 - Same as Experiment 2 except that for step 4 and 5, we compare the probability distribution of topics.

For example, assume we find three topics for the requirement data. REQ1 has a topic distribution of $\{ (0, 0.5), (1, 0.25), (2, 0.25) \}$ and UC1 has a topic distribution of $\{ (0, 0.25), (1, 0.25), (2, 0.5) \}$ and UC2 has a topic distribution of $\{ (0, 0.6), (1, 0.2), (2, 0.2) \}$, where topic distribution is a set of tuples and each tuple represents pair of the topic index and its weight. We now use the method of least squares to calculate the distance of these REQ/UC.

$$\text{For REQ1 and UC1, } (0.5 - 0.25)^2 + (0.25 - 0.25)^2 + (0.25 - 0.5)^2 = 0.125$$

$$\text{For REQ1 and UC2, } (0.5 - 0.6)^2 + (0.25 - 0.2)^2 + (0.25 - 0.2)^2 = 0.015$$

Thus UC2 is the better link for REQ1 because it is closer in similarity.

4.8 Validation and Acceptance Test

Development of standard accuracy calculation methods, which are necessary to effectively compare the performances of different algorithms, is mission-critical. We have tried various methods for testing and we will talk about the pros/cons of each method.

The following are some of the suggested methods. For all of the following methods, we assume that the test data is perfect, meaning that only good links are listed for each requirement and there are no missing links within the file.

🏠 Most naive accuracy measure:

1. For each REQ, check if one of the predicted links is in the set of existing links
2. If so, mark this requirement as 1, otherwise 0.
3. Take sum of these values for all requirements and take average.

It's computationally cheap to calculate this. However, the problem of this measure is that it is such an easy grading. It needs to be more strict to effectively measure the accuracy.

🏠 Simple accuracy measure:

1. For each REQ, Take top N predictions, where N is the number of existing links.
2. Calculate Accuracy = (# of correct predictions) / (# of all predictions) for each REQ.
3. Take average of that measure for all REQs.


It's also computationally cheap to calculate this. However, the problem is that it only considers good links.

🏠 Another simple measure using set theory concept:

1. Accuracy = $(L_e \cap L_p) / (L_e \cup L_p)$, where L_e is a set of existing links and L_p is a set of predicted links. [(Correct prediction) / (correct + wrong predictions)]
2. Again, take average of it

🏠 Measure using confusion matrix [3]:

1. For each REQ, calculate accuracy = $(TP + TN) / (TP + TN + FP + FN)$
2. Take average of them



Since we have the problem of not considering bad links, we developed another measure using confusion matrix. Creating a confusion matrix is an ideal approach because it can reveal lots of information about the model with just a simple graph. Important statistics that will be analyzed from the confusion matrix will be overall accuracy of the model, number of false positives, and number of true negatives. Analysing these statistics will tell us how we need to calibrate the model for better results, or if the approach needs to be rethought as a whole.

So far this is the most reliable method we can think of. However, the problem is that it takes significant human effort since we need to manually select really bad links, the links that seem very different from the requirement. For prototyping we should use one of the accuracy calculation measures listed above.

4.9 Cost Consideration

In the context of our project, there are not any cost considerations to take into account. All of the technologies we are using to build our tool (e.g. Python for the programming language, as well as libraries such as NLTK, Word2Vec, SpaCy, etc) are all free to use, so Collins Aerospace and our design team do not need to consider the costs of purchasing technologies in order to build our tool.

4.10 Possible Risks and Risk Management

4.10.1 Risks

This project is considered to be safety critical since the requirement information that it will ultimately be analyzing once implemented by Collins Aerospace is highly classified information. Therefore, the main risks of this project pertain to security. We must be sure that it is impossible for any individual outside the employees of Collins Aerospace to be able to access any information that is input into our tool (e.g. the documents containing details of requirements and use cases for a certain project) or output from our tool (e.g. the analysis of the given requirement trace and recommended additional use case links for any requirements).

We must also be sure that our analysis algorithms are implemented as accurately as possible. Collins Aerospace will be relying on this tool to analyze their requirement traces for projects that rely on these traces being extremely accurate, so one of the biggest risks in this project is a failure in one of our algorithms causing a good requirement link to be classified as a bad requirement link or causing a bad requirement link to be classified as a good requirement link, and having this requirement trace get carried forward in the project.

4.10.2 Risk Management

In terms of security risks, we will need to have further discussion with our contacts at Collins Aerospace in order to determine our mitigation strategy. Since Collins is a very large corporation that has developed many safety critical projects, it would make sense that they have a standard, or at least certain requirements, on how to implement their security software. They will either be able to provide us with guidelines on how to implement this ourselves, or may hand the project over to their cyber-security experts and implement these features internally.

While there is no guarantee that link classification will always be perfect, we are mitigating the risk of incorrect classifications by continuing to research different similarity algorithms and testing their accuracy against each other. Over the course of next semester, we will make a final decision of what algorithm we will use in our final implementation based on the results of our experiments.

4.11 Feasibility Analysis

The feasibility of the topic modeling approach cannot be completely determined as of the current point of the project. With that being said, it is the most reasonable approach that we have considered so far on the following terms:

4.11.1 Unsupervised Learning:

Topic modeling is an ideal solution due to the fact that it is under the category of unsupervised learning. Due to the randomness and variance of the requirement data between projects, there is a lack of solid features that could be picked to define a likelihood that a given link is good or bad. Using unsupervised learning will allow us to train a model which picks an appropriate set of topic distributions for any given directory no matter how they were written on for that project.

While this provides an interesting advantage for our project, it also creates concerns. Since we are not in control of how the model creates these topics, there is no way to know that it will be choosing topics in a way that models links between requirements. This is not something we will know until more testing is performed for this approach on Collins Aerospace requirement data.

4.11.2 Providing Topics/Probability Likelihoods:

Another positive aspect of using the topic modeling approach is how the model represents data. The topic model outputs probability distributions to topics for any given requirement we query. This allows us to easily suggest possible links according to the topic probability distribution output. It also allows us to easily compare two requirements probability distribution to determine whether they should be linked or not. Since we are defining the process for classifying links as good or bad, we can also create our own threshold of what should be classified as a good, bad, or even suspicious link. Overall, the outputs of the model will be easy for us to work with in terms of both classifications and suggestions.

4.11.3 Trainability:

This approach eliminates the problem of out of vocabulary words which we have run into with previous solutions such as word2vec and glove. Collins Aerospace requirements include flags and acronyms which are continuously introduced. When these are encountered, there is no way to train the models on words without extensive training data. The problem, however, is since these flags and acronyms were just created, there's no efficient way to train on them which renders them difficult to manage. The topic modeling approach does not rely on learning the meaning of individual words. Instead, it picks similarities in the provided data to define topics with, allowing it to eliminate the roadblock of out of vocabulary words.

4.11.4 Conclusion of Feasibility:

Over all, this approach seems feasible and may be promising in finding a solution towards analyzing Collins requirement data. Topic modeling allows for unsupervised learning, to account for the variance in data; has conveniently structured outputs for us to use in classification and suggestions; and also removes roadblocks such as out of vocabulary words. However there is some concern which will need to be tested for whether the approach can consistently and accurately model requirement data.

4.12 Project Proposed Milestones and Evaluation Criteria

Milestone 1: Research Machine Learning Algorithm for Requirement Traceability

Milestone 2: Experiment on Milestone 1

Milestone 3: Analyze the result obtained in Milestone 2

Milestone 4: Design the Algorithms and Interfaces for the primary requirements

Milestone 5: Develop the software with primary requirements

Milestone 6: Create Testing Modules

Milestone 7: Determine Secondary Requirements

Milestone 8: Implement secondary requirement

Milestone 9: Create testing modules for secondary requirements

Milestone 10: Create Testing Module for software with primary and secondary requirement.

Milestone 11: Code Review, software review, required instruction manual and videos for demonstration.

Milestone 12: Final review and presentation

4.13 Project Tracking Procedure

Our team is using the agile development methodology for this project with a two week sprint duration. In order to track progress on individual tasks, our team set up a Jira server. With Jira, we are able to create tickets for each task in the current sprint and assign those tasks to members of the group. We have also created a Git repository for version control purposes, where we all push experiments to our own individual branches and then merge our work to the master branch once we have finished implementing a certain functionality.

4.14 Test Plan

As the part of the project is related to the traceability. The traceability needs to be worked in forward, backward and also in bi-directional. Test Modules will be created for each of them which are introduced below.

4.14.1 Forward Traceability Test

As forward traceability test, the software needs to verify if it could map the higher level requirement only to a lower level requirement. Test cases will be created keeping in mind the example show below:

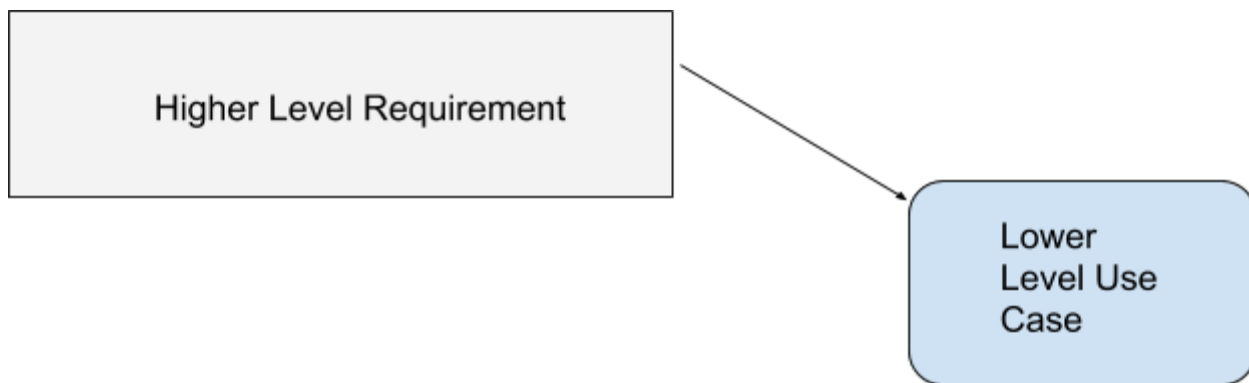


Figure 5: Test Case for Forward Traceability

Thus, it will test if the higher level is been correctly mapped with a lower level use case.

4.14.2 Backward Traceability Test

As backward traceability test, the software will need to verify if it could relate the lower level use case with its higher level requirement efficiently. Test cases needs to be created keeping in mind the example shown below:

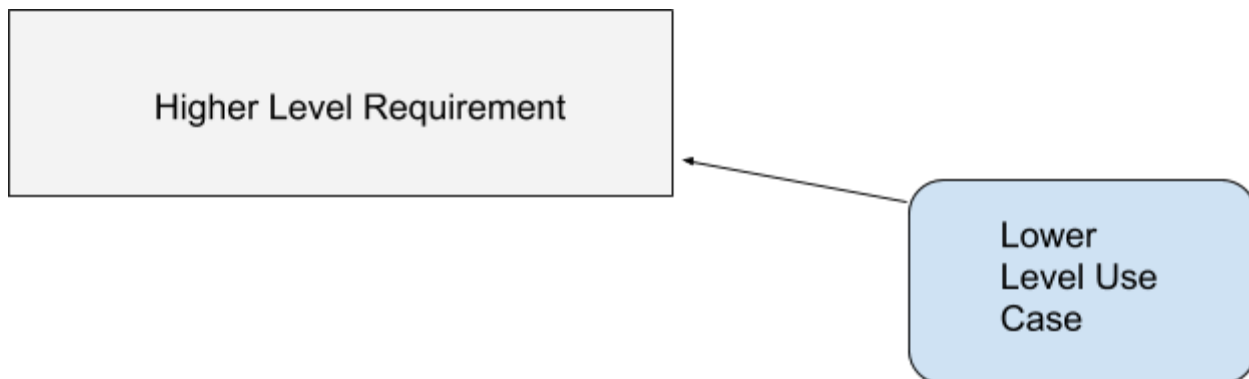


Figure 6: Test Case for Backward Traceability

Thus, a test case will be created to check if lower level use case traces back to its desired higher level requirement.

4.14.3 Bi-Directional Traceability Test

Bi-directional traceability test will test if both higher level requirement and the lower level requirement traces back to each other. Thus, test cases will be created keeping in mind the following scenario as shown below:

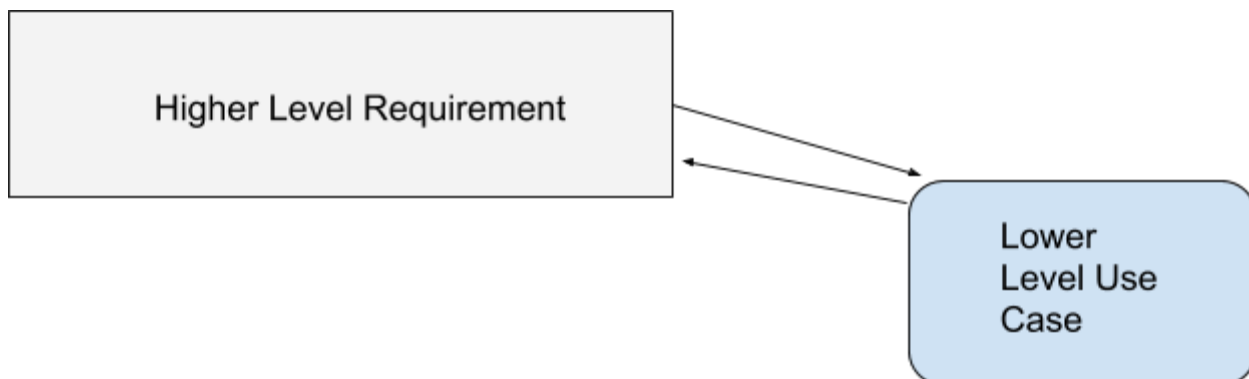


Figure 7: Test Case for Bi-Directional Traceability

Thus, a test case will be created to confirm that bi-directional traceability functionality.

4.14.4 Classification and Prediction Test

We will test the classifier by doing a series of 80/20 training/test data splits and generating confusion matrix on each run. An 80/20 training test split means that we will choose 80% of the data to train the model, chosen at random. Once the model is trained, the model will attempt to classify the rest of the 20% of data. We will monitor how the model classifies the 20% of test data and generate confusion matrix on it which gives us statistics on how the model is performing, including accuracy statistics.

5. Assessment of Proposed Solution

5.1 Summary of suggested approaches

We first came up with solutions using word2vec/WMD approach using a pretrained model. It turns out it is difficult since the pretrained model doesn't contain a lot of technical terms, acronyms, and flags that are in the requirements data. Retraining the model is difficult since the model has already learned many vocabulary and its connections. Another problem we faced is the fact that we have to deal with bad links. We then trained model on our own dataset but the result shows low accuracy. Then we figured that basically the problem is directly comparing texts. So we came up with a different approach that indirectly compare the text similarity, one of them is topic modeling. Heavy preprocessing is necessary to effectively find the text similarity. This is currently the state of our research project. We are actively researching topic modeling as well as other deep learning methods like recurrent neural network (RNN). We are also investigating the possibility of supervised learning, which also takes the existing links into account when training the model.

5.2 Assessment of Word2Vec and Word Mover's Distance

Word2Vec paired with Word Mover's Distance, as stated in section 4.7.2, proved not to be the best combination to solve our problem. Word Mover's Distance is build more so to identify rephrasings of sentences as similar, or even sentences with the general same meaning/context. However, this is not how Collins Aerospace writes their requirements. All requirements, whether classified as a good or bad link, returns within the same range of Word Mover's Distance. This tells us that the algorithms does not tell a difference between good and bad links by soley looking at the context of the requirements text. A better solution will find characteristics of good links and bad links and will create a classifier from those characteristics.

5.3 Assessment of GloVe

After implementing GloVe and experimenting on the small requirement dataset, the results were very similar to the experiments we performed that implemented Word2Vec with Word Mover's Distance. Relying on cosine similarity between vectors can be very effective in comparing individual words, but when comparing an entire sentence, a cosine similarity calculation of each word cannot effectively take into consideration the context or topic of the full sentence. Even after the removal of stop words (e.g. common words that appear in almost any English sentence such as 'a' or 'the'), the average similarity of good links and average similarity of bad links varied by a very insignificant amount, making the two classifications indistinguishable based on the similarity calculations produced from the implementation of GloVe. In our case, as well as in the study [5] referenced below, GloVe and Word2Vec performed similarly when computing sentence similarity, but were not considered sufficient. It may be that there is more preprocessing work to be done rather than just filtering out stopwords, or filtering out any word that is not a noun. GloVe allows you to define your own custom global vectors, so with more experimentation it is possible that we discover a way to utilize these custom definitions in order to more effectively classify links.

5.4 Assessment of topic modeling

Topic modeling experiment 1: The purpose of this experiment was to show the feasibility of topic modeling and it hopefully works better than existing approaches that directly compares the REQs/UCs. But it got to the problem that we are not sure how to compare two separate topic models. Topics are a set of words with weight. However, LDA doesn't tell the difference between, for example, 'user and 'users'. We can use some pre-trained word2vec model for this problem but it is going to be really complex process. For example, how do we decide which word to unify similar words? We eventually gave up on this approach.

Topic modeling experiment 2: This experiment was done as a response to experiment 2. Instead of making separate models for REQs and UCs we can just make one model. This way we don't need to worry about figuring out to compare separate topic models. This idea made it much easier to implement the algorithm and we were successful in making something working. Our result using naive accuracy measure shows that it's got accuracy of 57%.

Topic modeling experiment 3: This experiment was done as a response to experiment 3. The difference is that instead of taking the topic with the highest weight and compare the UCs using that topic, we compare the topic probability distributions. This shows better accuracy than topic model 2.

Other than all these topic modeling experiments we have done, we should definitely train bigger data and see the accuracy of our approaches.

6. Estimated Resources and Project Timeline

6.1 Estimated Resources

6.1.1 Personnel effort requirement

Task	Description	Effort Needed
Research Learning Algorithm	Research on algorithms that are available for sentence similarity	Most of the research done so far has turned out to be negative of what we wanted so far. Also, since none of the team member is highly educated on the machine learning aspect of software development. It is very important for each of the team member to understand and research on the algorithms efficiently
Experiment Learning Algorithm	Conduct experiments on the research related to the machine learning algorithms	The gensim library includes many of the algorithms that we will conduct the experiment on. However, there are few experiments that needs to be conducted out of the gensim library in order to make the algorithm more efficient and functionable as per the requirement.
Analyze the experiment	Once the experiment is been conducted, it should be analysed as per the client's requirement	Since the project has very strict and specific requirements, once we have seen how each algorithm works through research and experimentation, we will decided which algorithms best suit the client's needs.
Design Algorithms	After deciding on the best learning algorithm, we must design an algorithms that	There will be a considerable amount of attention given to this task. The algorithm designed

	analyzes the similarity computed by the learning algorithm in order to flag requirement links as good, bad or suspicious. The algorithm must also identify requirements that do not have any links and based on computed sentence similarity.	here will define the efficiency of the most of the system. With that been said, we want this to be as efficient as we can possibly make it. Once an algorithm is created, it may continue to be improved until the end of the project.
Test the Design Algorithm	Once the algorithm is been decided, testing the algorithm will take place	Implementation of the Designed Algorithms needs to be performed in python3.7 using necessary libraries. Once the implementation is done, testing on different test cases or using a junit or mockito needs to be performed.
Begin Developing Modules	Write Modules for the project	Implement modules irrespective of the size. Some of them are large and will be time consuming, however others will be pretty straightforward to implement
Test Modules	After implementation of Modules, thoroughly test it to ensure that it meets the expected requirements	Testing will require much attention during the development process. We will need to ensure that all the desired requirements by our clients are met. Thus, we must develop test cases for each of the requirement and analyze it thoroughly.
Insert Tested Modules into project structure	Once a module has been completed, we will merge it with other finalized modules in order to continue to add to the final build of the project.	This step is very straightforward, since it only requires pushing our work into the repository where we are storing the finalized project modules.

Table 1: Personnel Effort Requirement

6.1.2 Other resource requirement

Jira is used as an external resource for project management process. In order to host jira, we are using an apache web server based on Linux 16.0.1 hosted at Iowa State University ITS services. Apart from that, we are using number of libraries from python such as gensim, nltk, pandas, scikit learn, scipy, spacy, matplotlib, numpy and few built-in libraries. For the installation process of the external libraries supported by python, we use anaconda 3 as well as miniconda which is a package manager for dependencies.

6.1.3 Financial requirement

This tool is developed with python and open source software. Therefore, there will be no financial requirements to be met.

6.2 Project Timeline

Collins Aerospace has primary as well as secondary requirements. Most of the time in Spring semester was spent on research on primary requirements. We worked on researching, experimenting, analyzing and testing algorithms such as word2vec, word mover distance, glove, and topic modeling - LDA. We will be continuing research, experimentation and analysis of different algorithms as well in the second phase of the semester to improve the result. Along with that, we will be implementing the software as a whole with testing modules. Apart from that, we will also implement secondary requirements. Secondary requirements are still under review by Collins Aerospace. As per our client, he will be introducing it during the next semester. Thus, the timeline below with the gantt chart is tentative and has to be declared later based on the proposal of the secondary requirement.

7. Standards

7.1 Unethical Processes

No process that this team uses during the development of this project would be considered unethical by organizations such as IEEE. We have taken a number of their standards including the IEEE Standard for Software Reviews and Audits [10], and the Systems and Software Engineering Standards for Developing Information for Users [11] into consideration in order to guide our decision making throughout the process of development.

7.2 Team and Company Interactions

When we are holding a meeting with our contacts at Collins to demonstrate the progress made during our previous sprint, we structure those meetings based on the standards given in the IEEE Standard for Software Reviews and Audits [10] in order to guide our decisions on how to prepare for the meeting by establishing roles within our team for the meeting based on those defined within these industry standards such as the recorder who documents the discussion for later reference or the review leader who leads the discussion when demonstrating the progress our team has made over the sprint. In terms of internal interaction among team members, we are operating using the agile development methodology. This includes us having frequent stand up meetings, developing user stories and use cases, and conducting sprint meetings in order to lay out development goals for the upcoming two week sprint. When structuring all of these artifacts and conducting these meetings, we look to the Systems and Software Engineering Standards for Developing Information for Users in an Agile Environment [11] for guidance based on the industry standards.

7.3 Security

As previously mentioned, we are not at a point in this project yet where we have implemented any sort of security features, but the implementation of these features is a critical part of this project. We will need to ask Collins Aerospace if there are any standards they have for securing their software and will need to first consider that as a guideline. Additionally, we will refer to the IEEE Standards for Software Safety Plans [12] in order to educate ourselves about industry standards relating to development and maintenance of software systems.

8. Closure Material

8.1 Closing Summary

Requirement Tracing is important in developing reliable softwares, especially in large organizations such as Collins Aerospace. It helps in minimizing the efforts and errors by software engineers on writing huge test cases, higher and lower level requirements, and mapping each other to obtain a desired output. Collins Aerospace Artificial Intelligence for Requirement Analysis Tool will efficiently replace the current method where engineers would have to enter the requirements and test cases manually. Moreover, it will also reduces the human error caused during the process. Overall, completion of this project will provide Collins Aerospace to be one of the rarest companies to achieve artificial intelligence for requirement traceability.

8.2 References

- [1] Sorte, Bhagyashree W. et al. "Use of Artificial Intelligence in Software Development Life Cycle: A state of the Art Review." (2015).
- [2] Li, Zeheng and LiGuo Huang. "Tracing Requirements as a Problem of Machine Learning." (2018).
- [3] "Confusion Matrix." *Wikipedia*, Wikimedia Foundation, 4 Feb. 2019, en.wikipedia.org/wiki/Confusion_matrix.
- [4] "Latent Dirichlet Allocation." *Wikipedia*, Wikimedia Foundation, 24 Apr. 2019, en.wikipedia.org/wiki/Latent_Dirichlet_allocation.
- [5] Peirsman, Yves. "Comparing Sentence Similarity Methods." *NLP Town Blog | Comparing Sentence Similarity Methods*, 2 May 2018, nlp.town/blog/sentence-similarity/.
- [6] "A Beginner's Guide to Word2Vec and Neural Word Embeddings." *SkyMind*, skymind.ai/wiki/word2vec.
- [7] Ma, Edward. "Word Distance between Word Embeddings." *Towards Data Science*, Towards Data Science, 25 Aug. 2018, towardsdatascience.com/word-distance-between-word-embeddings-cc3e9cf1d632.
- [8] "Gensim: Topic Modelling for Humans." *Radim Řeháček: Machine Learning Consulting*, radimrehurek.com/gensim/models/ldamodel.html.
- [9] PyOhio. "Natural Language Processing in Python." *YouTube*, YouTube, 29 July 2018, www.youtube.com/watch?v=xvqsFTUsOmc.
- [10] "1028-2008 - IEEE Standard for Software Reviews and Audits." *IEEE*, Software & Systems Engineering Standards Committee, 2008, standards.ieee.org/standard/1028-2008.html.
- [11] "ISO/IEC/IEEE 26515:2018." *ISO*, 12 Dec. 2018, www.iso.org/standard/70880.html.
- [12] "1228-1994 - IEEE Standard for Software Safety Plans." *IEEE*, 1989, standards.ieee.org/standard/1228-1994.html.