

# **AI Based UML Diagrams Generator**

**K. A. D Oshada Kasun Kiringoda Arachchi**

**2021**



# **AI Based UML Diagrams Generator**

**A dissertation submitted for the Degree of Master of  
Computer Science**

**K. A. D Oshada Kasun Kiringoda Arachchi**  
**University of Colombo School of Computing**  
**2021**



## DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. This thesis has also not been submitted for any degree in any university previously.

Student Name: K. A. D Oshada Kasun Kiringoda Arachchi

Registration Number: 2018/MCS/047

Index Number: 18440474



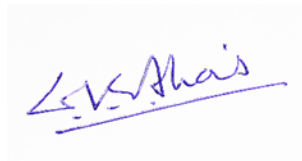
30/11/2021

Signature of the Student & Date

This is to certify that this thesis is based on the work of Mr. /Ms. K. A. D Oshada Kasun Kiringoda Arachchi under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by,

Supervisor Name: Prof G.K.A. Dias



\_\_\_\_\_  
Signature of the Supervisor & Date 30/11/2021

I would like to dedicate this thesis To My Beloved Parents....

## ACKNOWLEDGEMENTS

It gives me great desire to acknowledge all those who supported me during the course of this project.

First of all, my heartfelt gratitude goes out to Prof. Kapila Dias, my project supervisor and my project proposal approver for his support throughout the project.

To my loving parents and brother, for supporting and encouraging me to make this project a success.

To all the academic and non-academic staff at UCSC for their support, for allowing to continuing studies without any issue during the pandemic situation.

My colleagues at Yukon Software, for the corporation extended and the inspiration from the beginning.

But last not least all my colleagues from UCSC, who have always been there for me, encouraging me when I was down and strengthening me.

A very big and whole-hearted thank you to all of you. This would not have been possible if not for all of you.

## ABSTRACT

Unified Modeling Language (UML) diagrams are very useful to represent the business requirement of any proposed system and help to design a system from the end user's point of view. In the software analysis process, UML diagrams are drawn separately after gathering requirement, and the time wastage of drawing these diagrams is high with using current drawing tools due to the complexity of the business situation or the technical capabilities of the UML diagrams. Automated UML Diagram generation tools necessity can be identified with amount of time spent on requirement analysis and low quality of human analysis. The objective of this project is to provide an approach to generating UML use case and class diagrams from the functional requirement texts using natural language processing and machine learning.

Finding the key terms for class or use case diagram is reflected a classification task in machine learning. The stories narrate different attributes of a diagram and the task is to identify key terms for respective attributes. The use case diagram and class diagram have different set of attributes. classification model can be designed and pre-process the data for extracting the key terms such as actors, use case and classes from the requirement text. Furthermore, other text features like position and distance of text can be integrated to improve accuracy. To Identify Use case and class relationship sequence to sequence RNN model has introduced and it extracted key phrase or relation phrase among text, which also outperformed the conventional technique significantly. Improved strategies were proposed on this approach by substituting a sequence-to-sequence RNNs with conventional techniques.

The proposed system is capable of providing solutions to generating usecase and class diagram from the functional requirement text with using NLP and ML techniques. Diagram generating has used Plant UML tool with its plan text language where user interaction also enabled. Developed prototype is able to identify important actors, use cases and its relations with relationship type for use case diagrams. And also for class diagrams important classes and it's relationships can identify. If this project is domain specific most of element would have been identified. The developed final model has identified and generated the use case diagram and class diagram up to a considerable extent according to the given requirement text which gives a good idea about the business scenario.

Keywords: UML Diagram, Use case Diagram, Class Diagram, Natural Language Processing, Machine Learning, Classification, RNN, sequence-to-sequence RNN, Plant UML

# TABLE OF CONTENTS

DECLARATION.....	i
ACKNOWLEDGEMENTS .....	iii
ABSTRACT .....	iv
TABLE OF CONTENTS .....	v
LIST OF FIGURES .....	vii
LIST OF TABLES .....	viii
CHAPTER 1 INTRODUCTION.....	1
1.1 Motivation.....	1
1.2 Statement of the problem .....	1
1.3 Research Aims and Objectives .....	2
1.3.1 Aim .....	2
1.3.2 Objectives .....	2
1.4 Scope.....	2
1.5 Structure of the Thesis .....	3
CHAPTER 2 LITERATURE REVIEW.....	5
2.1 Literature Review .....	5
2.1.1 Natural Language Processing and Machine Learning.....	5
2.1.2 Derive Use case elements from requirement text.....	6
2.1.3 Derive Class Diagram elements from requirement text .....	8
2.1.4 Generate use case and class diagram.....	8
2.2 Presentation of Scientific Material .....	9
CHAPTER 3 PROBLEM ANALYSIS AND METHODOLOGY .....	15
3.1 Problem Analysis.....	15
3.2 Proposing Model/design .....	16
3.3 Methodology .....	17
3.4 Algorithmic design .....	19
3.4.1 Preprocess Functional requirement texts.....	22
3.4.2 Machine Learning Techniques to Identify Diagram Elements.....	23

3.5	Data acquiring methods .....	26
CHAPTER 4 IMPLEMENTATION .....		28
4.1	Identifying Use case and Class Diagram Entities with NLP and ML techniques ..	28
4.2	Identifying Use case and Class Diagram Relationships with NLP and ML techniques .....	32
4.3	Draw diagram with identified diagram elements .....	36
CHAPTER 5 EVALUATION AND RESULTS .....		39
5.1	Collecting Dataset .....	39
5.2	Dataset Preparation .....	41
5.3	Evaluation approach .....	42
5.4	Evaluation and Results.....	44
5.4.1	Identifying use case diagram element from requirement text .....	44
5.4.2	Identifying class diagram element from requirement text.....	52
5.4.3	Generating usecase and class Diagram.....	58
CHAPTER 6 CONCLUSION AND FUTURE WORK.....		65
6.1	Conclusion .....	65
6.2	Future Work.....	67
REFERENCES .....		I
Appendix A .....		V



## LIST OF FIGURES

Figure 2.1 ML Model prediction flow .....	9
Figure 3.1 Proposed Solution Design .....	17
Figure 3.2 Process flow with NLP And ML Model .....	19
Figure 4.1 Sample JSON output format .....	36
Figure 5.1 Sample dataset format with functional requirement .....	41
Figure 5.2 Confusion matrix for the case of binary classification .....	43
Figure 5.3 Program generated language code for usecase diagram .....	59
Figure 5.4 Edited language code for usecase diagram .....	59
Figure 5.5 Actual usecase diagram according to the functional text.....	60
Figure 5.6 Generated usecase diagram using Plant UML tool.....	61
Figure 5.7 Program generated language code for class diagram.....	62
Figure 5.8 Edited language code for class diagram.....	62
Figure 5.9 Actual usecase diagram according to the functional text.....	63
Figure 5.10 Generated class diagram using Plant UML tool .....	63

## LIST OF TABLES

Table 2.1 Research gaps on deriving use case diagram elements .....	9
Table 2.2 Research gaps on deriving class diagram elements.....	10
Table 2.3 Existing research evaluation.....	11
Table 3.1 Sample dataset.....	20
Table 5.1 Test data snippet for actor in functional text .....	45
Table 5.2 Model Predicted results snippet for identified actors.....	45
Table 5.3 Evaluation metric result for actors .....	46
Table 5.4 Test data snippet for usecase in functional text .....	46
Table 5.5 Model Predicted results snippet for identified usecases.....	47
Table 5.6 Evaluation metric result for usecases .....	47
Table 5.7 Data snippet from preprocessing stage data separation.....	48
Table 5.8 Test data snippet for relationship involved entities in functional text.....	49
Table 5.9 Model Predicted results snippet for identified usecase relationship involved entities .....	50
Table 5.10 Test data snippet for relationship Type in functional text.....	51
Table 5.11 Model Predicted results snippet for identified usecase relationship type.....	51
Table 5.12 Test data snippet for classes in functional text.....	52
Table 5.13 Model Predicted results snippet for identified classes .....	53
Table 5.14 Evaluation metric result for classes.....	53
Table 5.15 Data snippet from preprocessing stage class relationship data .....	54
Table 5.16 Test data snippet for class relationship involved entities in functional text.....	55
Table 5.17 Model Predicted results snippet for identified class relationship involved entities ... .....	56
Table 5.18 Test data snippet for class relationship type in functional text .....	57
Table 5.19 Model Predicted results snippet for identified class relationship type .....	58

# CHAPTER 1

## INTRODUCTION

In the Software development industry Requirement gathering and Design phase plays an important role in software development lifecycle (Whitney, E., CODE Magazine 2020). Depending on the time taken to these phases whole project duration will be affected since different parties who are playing different kind of role need to provide their feedback and also some authorities need to be agreed whether the Design would be satisfied with the requirements. More importantly particular client needs to be satisfied with the Design with gathered requirements.

### 1.1 Motivation

Creating UML diagrams is a vitally important and time-consuming task which both requirements and design phases in software Development. These diagrams like use-case Diagram and class diagram will consider as a transition between the two phases. Use case diagram consider as the one of the most used functional modeling techniques that use in the software development process and the main core of Object-Oriented analysis and design where other models are derived is class diagram.

From the initial survey carried out by Madanayake R. S., (2019) in his research stated that “Class diagrams, ER Diagrams, User stories and use case models were the most popular diagrams used as well as where the most Duplication of Work occurred, according to the data analysis conducted by the researchers.”

Since most of stakeholders are not aware on this diagram technique and not have capability of understanding the particular diagram with comparing to the business requirement may lead to consume more time on creating these diagrams against business requirements. Hence, reducing the time taken to create the use-case diagram against the business requirements is the motivation behind this project.

### 1.2 Statement of the problem

When drawing these UML Diagrams Requirement engineer or Business analyst need to put lot of work with gathering and analyzing business requirements and also good amount of time will waste due to this time taken process. UML Diagrams will draw separately after gathering requirements and wastage of time will increase when drawing these Diagram using current diagram tools according to particular business requirement due to complexity of business scenario or technical capability of UML diagram concepts. Automated UML Diagram

generating tools necessity can be identified with the how much time would spend on analyzing business requirements and also the low quality of the human analysis.

### **1.3 Research Aims and Objectives**

This project proposes an approach to facilitate use case and class diagram extraction from textual requirements using AI techniques.

#### **1.3.1 Aim**

The very first phase of the software development life-cycle is Requirement engineering. This phase, the requirements should be able to translated from the client language to the Developer's language. Usecase diagram and class diagram are two different approaches used to describe the functional requirements. Since the requirements can be written in natural language, Natural Language Processing Techniques (NLP) and Machine learning (ML) techniques would be able used to extract the information. This proposed model aiming to provide an approach to generate the UML usecase diagrams and class diagrams from the requirements text provided, using natural language processing and Machine learning.

#### **1.3.2 Objectives**

Following are the objectives that have achieved over the span of the research project:

- Identification and analysis most popular UML techniques and investigation of the existing techniques, practices and tools about drawing UML use case and class diagrams.
- Investigation of Natural language processing and Machine learning in the context of identifying use case and class diagram elements.
- Development of an AI based prototype for generate of use case and class diagram by analyzing the requirements.
- Identification of an appropriate evaluation mechanism to evaluate the solution.

### **1.4 Scope**

This study will mainly focus on providing solution to generating Usecase diagram and Class diagram against the particular business requirement. system will read and understand the business requirement using Natural Language Processing and Machine

learning and identifying entities and relationships on that for generate use case diagram. And also, classes and relationship between classes will also be identified to generate class diagram. Then system will show generated diagram with including user interacting feature where user can add additional element and add changes to use case or class diagram or edit existing element and its attribute or relations. There after user can confirm to generate both use case and class diagram respectively. The proposed system is limited only in reducing the time taken to draw use-case and class diagrams against the business requirements in software developments projects.

## **1.5 Structure of the Thesis**

### **Literature review**

With this chapter our aim to identifies, evaluates and synthesizes the relevant literature within a UML modeling, Natural Language Processing and Machine Learning field of research. It will illuminate how UML modelling and Artificial Intelligent techniques knowledge has evolved within the field as highlighting what has already been done and what is emerging, what is generally accepted and also what is the current state of thinking on the topic of UML modelling. In addition to these literature review chapter will identify a research gap such as under-researched or unexplored areas and articulates how this kind of research project would addresses this gap.

### **Methodology**

In this chapter need to answer these questions as how did I do this research and why did I do it that way. This will cover not only the methods used to collect and analyze relevant business requirement data, but also the theoretical approach and relevant techniques, framework used in the term of Natural Language Processing and Machine learning field. This would inform both the choice of methods and the approach to interpreting those data towards generating UML use case diagram and class diagram.

### **Evaluation and results**

In this chapter the results of the research project are presented and discussed evaluation with reference to the aim of the study, which was to generating UML diagrams against Business Requirements Using Natural language processing

and Machine learning. The two sub-aims - the first to find the most appropriate NLP and ML techniques towards identifying use case and class diagrams elements from business requirement, and the second to generate diagram using the diagram generating tool form the main comparisons in the evaluation. Evaluation will be done using a case study and solution given in a text book with the project output and also evaluation can be done with the help of domain experts in order to evaluate the efficacy of the provided solution.

### **Conclusion and future works**

This chapter would present the conclusions of the research described in this project thesis. The aim and objectives of this research, will be outlined in the Introduction chapter, and those are reviewed and their achievement addressed. It summarizes the focus of the Natural language processing and Machine learning approach on generating UML diagrams against Business Requirements, and draws conclusions from the discussions and results in the previous chapters. Proposals for future work indicated by the research are suggested.

## **CHAPTER 2**

### **LITERATURE REVIEW**

In this chapter our aim is to identify, evaluate and synthesize the relevant literature within a UML modeling, Natural Language Processing and Machine Learning field of research. It will illuminate how UML modelling and Artificial Intelligent techniques knowledge has evolved within the field as highlighting what has already been done and what is emerging, what is generally accepted and also what is the current state of thinking on the topic of UML modelling. In addition to these literature review chapter will identify a research gap such as under-researched or unexplored areas and articulates how this kind of research project would address this gap.

#### **2.1 Literature Review**

In this section will explore how UML modeling evolved with Natural Language Processing and Machine Learning field of research with identifying existing research that has been done so far.

##### **2.1.1 Natural Language Processing and Machine Learning**

In recent past years, different approaches have studied for the natural language requirements transformation into UML diagrams, from that few researchers have specifically focused on both usecase and class diagram using the Natural language processing techniques along with machine learning techniques. User requirement analysis can be an information extraction application of Natural Language Processing. It will be identified specific semantic elements in the business requirements entered in textual form as mentioned in Bhagat, S., Kapadni, P., Kapadnis, N., Patil, D. and Baheti, M. (2012). Therefore, this proposed system will come under the information extraction over the Information Retrieval and also in Questioning and Answering Tasks in NLP Moldovan, Dan & Surdeanu, Mihai. (2002).

In natural language processing several approaches were used to extract the information from given text in Barba, P., Lexalytics, (2020).

- Sentence splitting  
With this approach all text will split into sentences.
- Lexical Analysis  
This would get the split sentences and tokenize the sentences into words.
- Syntax analysis

This approach will receive the tokens generated by lexical analysis and applies a rule based or machine learning approach to generate and outputs parts of speech in a given text.

- **Word chunking**

With using this NLP approach, we can derive usecases from the input business requirement text. That will mostly be identify noun phrases, and verb phrases and also propositional phrases using tokenized text and POS tags.

Machine learning would be used to identify meaningful use cases, classes and identify relationship types in use case and class diagram. And also, we would use this machine learning techniques to identify multiplicities in the relationship of class diagram. Using machine learning techniques, we can improve the accuracy level of identifying above mention elements in both use case diagram and class diagram in a better way even though Rule based method also exists there as an alternative option.

Fig. 2.1 shows the text classification flow to be considered when applying machine learning techniques.

C. R. Narawita, K. Vidanage (2017) introduces a technique for generating UML model as use case and class diagram. This mainly focused on the design phase of a software. In this proposed system will extract usecase and class diagram elements with including relationships and also user interaction were involved to do necessary changes. They were used classification model to identify relationships and they suggested to use regression and they believe that would be a better approach compared to classification. And also reducing user interaction on this proposed system would be a good improvisation.

### **2.1.2 Derive Use case elements from requirement text**

This section will briefly review the major research efforts focused on the application of natural language processing technologies to gain insight from business needs to generate usage and class diagrams.

Some commercial products that have representing usecase models have been developed such as visual UML, Rational rose, smart Draw, MS Visio etc. Bajwa,I. S. & Choudhary,M. A. (2006). Some advanced tools have also been suggested to automate software engineering activities, which will be more complex than providing advice on drawing possibilities and other diagrams.



Hamza, Z. and Hammad, M., (2019) In this article, he proposes an approach to generate the use case diagrams from the business requirements using natural language processing techniques. This proposed approach consists of a few steps to proceed with the requirements. Starting with the filtering, the text errors and also went through natural language processes all the way to the generation of the particular use case diagrams.

Osman, M.S., Alabwaini, N.Z., Jaber, T.B. and Alrawashdeh, T. (2019) create a new approach which will focus on increasing precision of the technique which user requirements convert it to UML diagrams with reducing time of generating the usecases of requirement text written in natural language. This will also find solutions to some of the problems in current technologies as people need a smart and accurate solution to meet their needs with saving time and also the reliability increasing of the reliance on software.

Elallaoui, M., Nafil, K. and Touahni, R., (2018) present an approach were an Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques. In there they propose a process of transforming user stories into use cases and for that they used natural language processing techniques. But in this solution, it's not supported for exclude or include relationships between usecases and also it does not support sentences which will contain more than one compound noun. The limitations of this paper are derived associations from complex sentences, which will mostly require exclude or include relationships between usecases and this type of relationship not solved in this paper. And also, this does not resolve sentences containing more than one compound noun. The paper also does not mention generalizations and specializations and usage opportunities between actors and actresses and will not focus on this solution but in their future work.

Pereira, A., (2018) has proposed a work as using NLP to generate user stories from software specification in natural language and in there she proposed the user story generation approach and tool to simplify all the work that eliciting user stories required for software development.

Vemuri, S., Chala, S. and Fathi, M., (2017) Attempts to approach the problem of automation of the requirements analysis process automation process due to the text format of the requirements. Use probabilistic technology to identify usecases and actors. This has yielded promising results, and they state in their paper that this approach, which fully automates the analysis phase, will be further enhanced.

Further research analysis with research gaps on deriving use case elements to generate use case diagram can be found in Table 2.1 in 2.2 section.

### **2.1.3 Derive Class Diagram elements from requirement text**

This section will briefly review the leading research efforts focusing on the application of natural language processing to gain knowledge from the business requirement to generate a class diagram.

Nasiri, S., Rhazali, Y., Lahmer, M. and Chenfour, N. (2020) in this paper they discuss how to generate a class Diagram presented in the XMI file from the specifications given in the user stories. Improvements such as quoting the cardinalities of class diagram associations and applying artificial intelligence to generate new rules, especially for aggregation relationships, can have better results.

Jaiwai, M. and Sammapun, U. (2017) According to the article, the proposed method is to process the cable requirements written in Thai to extract UML class diagrams using natural language processing techniques, and the extraction of UML class diagrams is based on translation rules for identifying classes and attributes of these classes from the requirement text provided.

Adhav, V., Ahire, D., Jadhav, A. and Lokhande, D. (2015) in this paper, they used Natural Language Processing and domain ontology techniques to support class diagram extraction from human language requirements to find four types of relationships as generalization, association, composition / aggregation, and dependency. Inhere this Unable to identify one-to-one, one-to-many, and many-to-many relationships.

Kumar, S.K. (2014) The proposed system would allow developers to create a UML class model using software specifications using a standard configuration of natural language processing technologies and class diagrams, and additionally classify the relationship between classes.

More, P. and Phalnikar, R. (2012) This proposed method extracts UML diagrams from text requirements using natural language processing (NLP) and domain ontology technologies, as well as finding basic concepts and their connections and extracting UML diagrams.

Further research analysis with research gaps on deriving class diagram elements from requirement text to generate class diagram can be found in Table 2.2 in 2.2 section.

### **2.1.4 Generate use case and class diagram**

The output of this system is a PlantUML diagram where the project is to generate a use case and a class diagram using actors, classes, use cases, class attributes, usecase relationships and class relationships. There are many CASE tools for drawing the UML usecase and class diagrams. Visual paradigm, StarUML and Rational Rose are some of the famous software tools currently using in the industry. This project will use an integrated solution for the PlantUML as

it is easy to generate language code after deriving the use case elements and class diagram elements rather than using other tools (PlantUML, n.d.).

## 2.2 Presentation of Scientific Material

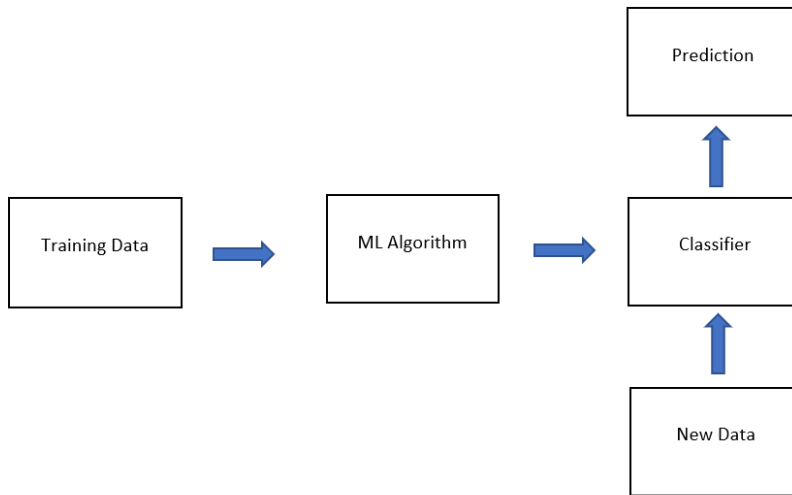


Figure 2.1 ML Model prediction flow

Figure 2.1 showing the Machine learning models prediction flow with using particular ML algorithm.

Table 2.1 Research gaps on deriving use case diagram elements

Research Topic	Deficiencies in Research
Identifying use cases	<ul style="list-style-type: none"> <li>Identifying meaningful and not meaningful use cases (Hamza, Z. and Hammad, M., 2019; Elallaoui, M., Nafil, K. and Touahni, R., 2018).</li> </ul>
Identifying Associations	<ul style="list-style-type: none"> <li>Identifying include relationships between use cases from complicated sentences (Osman, M., Alabwaini, N., Jaber, T. and Alrawashdeh, T., 2019; Elallaoui, M., Nafil, K. and Touahni, R., 2018).</li> <li>Identifying exclude relationships between use cases from complicated sentences (Osman, M., Alabwaini, N., Jaber, T. and Alrawashdeh, T., 2019; Elallaoui, M., Nafil, K. and Touahni, R., 2018).</li> </ul>

Identifying generalization	<ul style="list-style-type: none"> <li>• Identifying actor generalization (Narawita, C &amp; Vidanage, K. 2016; Elallaoui, M., Nafil, K. and Touahni, R., 2018).</li> <li>• Identifying use case generalization (Narawita, C &amp; Vidanage, K. 2016; Elallaoui, M., Nafil, K. and Touahni, R., 2018).</li> </ul>
----------------------------	---

Table 2.2 Research gaps on deriving class diagram elements

Research Topic	Deficiencies in Research
Identifying multiplicities	<ul style="list-style-type: none"> <li>• In a relationship in class diagram identifying <ul style="list-style-type: none"> <li>○ one to one</li> <li>○ one to many</li> <li>○ many to many</li> </ul> </li> </ul> <p>relationships accurately (Narawita, C &amp; Vidanage, K. 2016; Adhav, V., Ahire, D., Jadhav, A. and Lokhande, D. 2015).</p>
Identifying class attributes and methods	<ul style="list-style-type: none"> <li>• Complete selection of class attributes and methods by considering the frequency of words in the document (Nasiri, S., Rhazali, Y., Lahmer, M. and Chenfour, N. 2020; Kumar, S.K. 2014).</li> </ul>
Identifying Associations	<ul style="list-style-type: none"> <li>• Extract and Identify association classes from requirements (Nasiri, S., Rhazali, Y., Lahmer, M. and Chenfour, N. 2020; Jaiwai, M. and Sammapun, U. 2017).</li> </ul>
Identifying Generalization	<ul style="list-style-type: none"> <li>• Identify class generalization relationships (Kumar, S.K. (2014; More, P. and Phalnikar, R. 2012).</li> </ul>
Identifying Composition	<ul style="list-style-type: none"> <li>• Identify class composition relationships (Nasiri, S., Rhazali, Y., Lahmer, M. and Chenfour, N. 2020; Adhav, V., Ahire, D., Jadhav, A. and Lokhande, D. 2015; More, P. and Phalnikar, R. 2012).</li> </ul>

Table 2.3 Existing research evaluation

Research Title	What have done	Future works/ to do
UML generator – An Automated system for model driven development (Narawita, C & Vidanage, K. 2016)	<ul style="list-style-type: none"> <li>✓ Automate UML diagrams from the analyzed requirement text using NLP.</li> <li>✓ Extract usecase and class diagram elements</li> <li>✓ Relationships also identified</li> <li>✓ User will interact to do changes in extracted elements including relationship types.</li> </ul>	<ul style="list-style-type: none"> <li>• Used classification to identify relationships, suggested regression could be a good approach. (Weka vote algo has used.)</li> <li>• Reduce user interaction</li> </ul>
Generating UML Use Case Models from Software Requirements Using Natural Language Processing (Hamza, Z. and Hammad, M., 2019)	<ul style="list-style-type: none"> <li>✓ Provides access to generate UML usage diagrams from the Requirements using natural language processing.</li> </ul>	<ul style="list-style-type: none"> <li>• Future work is to automate the chunking process with the ability to deal with non-formatting requirement texts that make NLP processing easier in the approach.</li> <li>• Include, extends, relationships</li> </ul>
Generate use case from the requirements written in a natural language using machine learning (Osman, M., Alabwaini, N., Jaber, T. and Alrawashdeh, T., 2019)	<ul style="list-style-type: none"> <li>✓ This technology focuses primarily on increasing accuracy and reducing the time spent on generating systems that usecase text written in natural language.</li> </ul>	<ul style="list-style-type: none"> <li>• Identifying associations from complicated sentences such as include or exclude relationships between use cases</li> </ul>
Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques (Elallaoui, M., Nafil, K. and Touahni, R., 2018)	<ul style="list-style-type: none"> <li>✓ transform user stories into use cases and take the advantage from all the work done in the transformation process of the models according to the MDA approach.</li> <li>✓ sets of user stories generate the UML use case diagram automatically.</li> </ul>	<ul style="list-style-type: none"> <li>• include or exclude relationships between use cases not yet supported.</li> <li>• in their future work relationship type, such as generalization and specialization between actors and use cases will be addressed.</li> <li>• resolve sentences which containing more than one compound noun</li> </ul>
A Framework using NLP to automatically convert User-Stories into Use Cases in Software Projects (Azzazi, A. 2017)	<ul style="list-style-type: none"> <li>✓ correctly detected actors and use cases from user stories.</li> </ul>	<ul style="list-style-type: none"> <li>• Relationships didn't identify.</li> </ul>

<p>A Proposed Architecture for Automated Assessment of Use Case Diagrams (Vachharajani, V. and Pareek, J. 2014)</p>	<ul style="list-style-type: none"> <li>✓ This is for assessment of use case diagram where teacher will include problem statement and generate use case diagram then for particular problem statement student's usecase diagram will be evaluated.</li> </ul>	<ul style="list-style-type: none"> <li>• This will not draw usecase according to the input but evaluate and assess the drawn diagrams on particular format.</li> </ul>
<p>An Automated Use Case Diagrams Generator from Natural Language Requirements (Zakarya, M., Alqaralleh, B., Alemerien, K., Malek, Z., Alksasbeh and Alramadin, T. 2017)</p>	<ul style="list-style-type: none"> <li>✓ Read and performing a full analysis of the user requirements provided in the English language text.</li> <li>✓ It can also generate use case diagrams.</li> </ul>	<ul style="list-style-type: none"> <li>• This does not have the ability to reuse other existing use cases with using include, extend and generalize relationships</li> </ul>
<p>An Automated Tool for Generating UML Models from Natural Language Requirements (Deeptimahanti, D. and Babar, M., 2009)</p>	<ul style="list-style-type: none"> <li>✓ Generating the UML models like the Usecase Diagram, Analysis class model, Collaboration diagram and Design class model from the natural language requirements using more efficient Natural Language Processing (NLP) tools.</li> <li>✓ using available NLP tools with syntactic reconstruction rules to extract required OO artifacts like actors, usecases, classes, operations and attributes.</li> <li>✓ Identified actors, usecases and their association between actors and usecases.</li> <li>✓ generate use-case and class models from Natural Language requirements and collaboration and design class models from Use-case specifications along with proper relationships.</li> </ul>	<ul style="list-style-type: none"> <li>• Extending this function to automatically generate state chart diagrams for testing class models without the need for code generation.</li> </ul>
<p>Towards a Generation of Class Diagram from User Stories in Agile Methods (Nasiri, S., Rhazali, Y., Lahmer, M. and Chenfour, N. 2020)</p>	<ul style="list-style-type: none"> <li>✓ generates a class diagram, presented in XMI file format, and from specifications which are presented in user story wise.</li> </ul>	<ul style="list-style-type: none"> <li>• cardinalities of associations in the class diagram.</li> <li>• test criteria with user stories to build UML</li> </ul>

		<p>diagrams dynamically such as the Activity diagram.</p> <ul style="list-style-type: none"> <li>• Apply artificial intelligence techniques to generate new rules, importantly to detect aggregation relationships in between classes.</li> </ul>
<p>Extracting UML Class Diagrams from Software Requirements in Thai using NLP (Jaiwai, M. and Sammapun, U. 2017)</p>	<ul style="list-style-type: none"> <li>✓ This has processed the requirements written in Thai language to gain UML class diagram using the natural language processing techniques.</li> <li>✓ class diagram extraction based with the transformation rules which would identify classes and it's attributes from requirement text.</li> </ul>	<ul style="list-style-type: none"> <li>• identifying methods and relationships from requirements to generate class diagram completely.</li> </ul>
<p>Generation of UML Class Diagram from Software Requirement Specification Using Natural Language Processing (Kumar, S.K. 2014)</p>	<ul style="list-style-type: none"> <li>✓ Semi-automated designers can assist in the development of a UML class model from software specifications using natural language processing technologies.</li> <li>✓ class diagram in a standard setting and further identifying the relationship in between classes.</li> </ul>	<ul style="list-style-type: none"> <li>• Automate the selection of classes, attributes, and methods by considering the frequency of words in the document.</li> <li>• Identify different kind of modules and packages in the system.</li> </ul>
<p>Generating UML Diagrams from Natural Language Specifications (More, P. and Phalnikar, R. 2012)</p>	<ul style="list-style-type: none"> <li>✓ from textual requirements with using the natural language processing (NLP) and Domain Ontology techniques extracting the UML diagrams.</li> <li>✓ finding basic concepts and its relationships, and extract the UML diagram</li> </ul>	<ul style="list-style-type: none"> <li>• User interaction involving for add, delete and renaming classes with its relationships in the class diagram.</li> <li>• Allow the user to view, add, modify, organize concepts and relationships. The user can easily add new concepts, change the concept type and add new relationships.</li> </ul>
<p>Class diagram extraction from textual requirements using Natural language processing (NLP)</p>	<ul style="list-style-type: none"> <li>✓ Natural Language Processing and domain ontology techniques to support the extraction of class diagram from</li> </ul>	<ul style="list-style-type: none"> <li>• Unable to identify one to one, one to many, and many to many relationships</li> </ul>

techniques (Ibrahim, M. and Ahmad, R., 2010)	Natural language requirements. ✓ find four types of relationships: Generalization, Association, Composition/ aggregation and dependency.	
Semi-automatic Generation of UML Models from Natural Language Requirements (Deeptimahanti, D.K. and Sanyal, R. (2011)	✓ to guide developer in generating UML based analysis semi-automated technique has used in here and design models from normalized Natural Language requirements. ✓ capable to visualize UML diagrams in any UML modeling tool which has XMI import feature init.	<ul style="list-style-type: none"> <li>• Multiplicity among classes in class diagram is still need to be addressed in future work.</li> <li>• Generation of state diagram diagrams from use cases to test class models without the need to generate code, so that the test cases are requirement-based to test the behavior of the system.</li> </ul>

Table 2.1 and Table 2.2 shows the research gaps on deriving usecase and class diagram elements. Table 2.3 shows the evaluation of existing research where similar works done.

This chapter mainly focused on finding the best approaches, concepts and techniques to provide a solution to the problem domain. The chapter started by highlighting the existing researches done on this domain according to the different techniques in the field of NLP and ML. and also with the diagram generating techniques. Then presentation of scientific material also provided in here with research gaps on deriving use case diagram elements and class diagram elements. In the end detailed existing research evaluation has shown on the table.



## **CHAPTER 3**

### **PROBLEM ANALYSIS AND METHODOLOGY**

In this chapter focused on answering to the questions as how did the research was done and why it was done in that way. This will cover not only the methods used to collect and analyze relevant business requirement data, but also the theoretical approach and relevant techniques, framework used in the term of Natural Language Processing and Machine learning field. This would inform both the choice of methods and the approach to interpreting those data towards generating UML use case diagram and class diagram.

#### **3.1 Problem Analysis**

Generally, requirements are split into two types as Functional and Non-functional requirements (Guru99.com. 2019). Requirements that the client or vendor specifically demands as basic facilities is the Functional requirements and this will be the basic requirements stated by client/vendor. And also, these functional requirements will help to capture the intended behavior of the proposed system (Guru99.com. 2019). Non-functional requirements will define quality attributes of a suggested system and represent various standards that can be used to identify the operations of a suggested system (Guru99.com. 2019).

Representing the functional requirements in any suggested system use case diagram will be very helpful and will help in designing a system from the end user perspective (www.aha.io. n.d.). The use case diagram shows the dynamic aspects of a system with internal and external interactions. This is an effective technique which will help to communicate the behavior of the system in the terms of the client by specifying all the behaviors of the system visible from the outside (Visual-paradigm.com. 2019).

As a design phase, the first step will be to move from the problem domain to the solution domain. The objective of this phase is to draft the solution of the problem specified in the gathered requirements. Requirements will indicate as what is needed, design would take it towards how to satisfy those needs (Computer Notes. 2013). Class diagrams are defined as static diagrams, and the static view of need is represented by its attributes and functions. They can also be mapped directly with object-oriented source code (Holland, K. 2018).

In the Requirement gathering and design phase will be useful to avoid misunderstanding regarding the particular software system with involving the users. In order to do this most requirement engineers, need to analyze requirements and come out with UML Models. According to the initial investigation carried on during the preparation of this proposal we identified that, according to the surveys done by various researcher in software industry most popular UML models are the use case diagrams and class diagrams (Madanayake R. S., 2019).

When drawing these UML Diagrams Requirement engineer or Business analyst need to put lot of work with gathering and analyzing business requirements and also good amount of time will waste due to this time taken process. UML Diagrams will draw separately after gathering requirements and wastage of time will increase when drawing these Diagram using current diagram tools according to particular business requirement due to complexity of business scenario or technical capability of UML diagram concepts. Automated UML Diagram generating tools necessity can be identified with the time spent on analyzing business requirements and the low quality of the human analysis.

### **3.2 Proposing Model/design**

This study will focus on providing solution to generate Use-Case diagram and Class diagram against the particular business requirement. system will read and understand the business requirement using Natural Language Processing and Machine learning and identifying entities and relationships on that for generate use case diagram. And also, classes and relationship between classes will also be identified to generate class diagram. Then system will show generated diagram with including user interacting feature where user can add additional element and add changes to use case or class diagram or edit existing element and its attribute or relations. There after user can confirm to generate both use case and class diagram respectively. The proposed system is limited only in reducing the time taken to draw use-case and class diagrams against the business requirements in software developments projects.

According to the investigation carried out almost all works that have found was related to requirement text transform to one particular model diagram either use case or class diagram. But in this proposed system will review existing NLP and Machine learning techniques which have been applied to user stories or business requirements and will investigate how well they meet the requirement of text to model transformation.

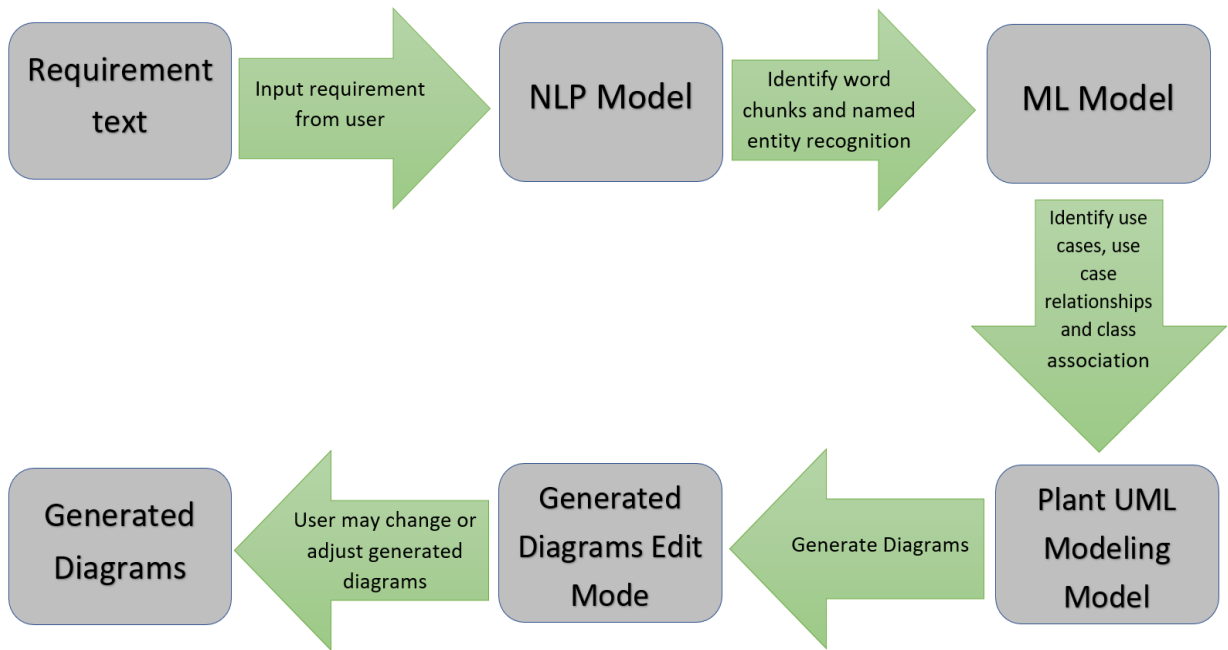


Figure 3.1 Proposed Solution Design

As showing in the Figure 3.1 proposed solution design, user can upload a text file containing a scenario or simply copy and paste the text inside. The system extracts word fragments and named entities using the NLP module. Subsequently, the ML module will identify and extract the usecases, actors, classes, associations according to the ML algorithms which have implemented. Plant UML modeling draws the use case and class diagram. The system allows editing of the generated diagrams and the user can modify and adjust the use cases, actors, classes, attributes and all types of relationships. Therefore, the output is a highly customizable output that the user will be able to achieve the desired output according to the business requirements provided.

### 3.3 Methodology

Text data contains a lot of information, but not all of it matters. We can search for names; others like to get specific links between these named entities. Our goal varies according to our needs. Sometimes we have to go through all the legal documents to find a legal precedent to validate your current case or you may need to go through all the research material to find relevant information on how to cure a disease. There are many other examples such as resume harvesting, media analysis, email analysis.

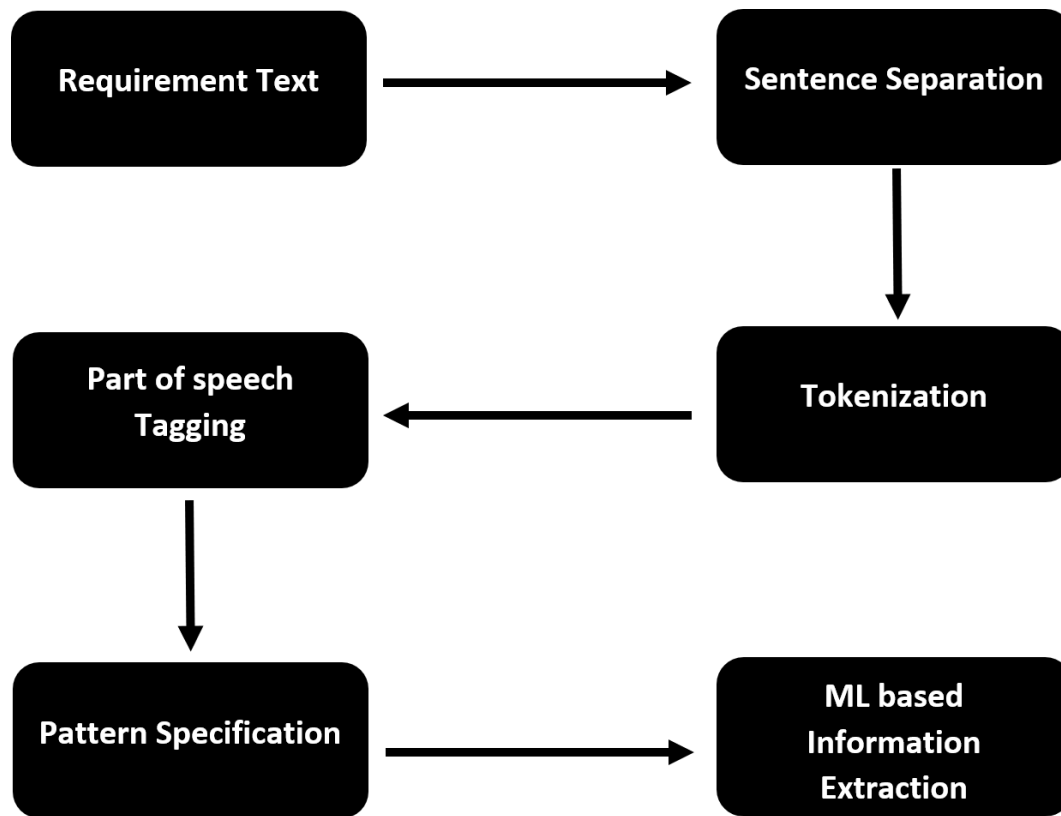
Imagine when we have to manually go through all the textual data and extract the most relevant information. Certainly, it is an uphill struggle and you might even end up skipping important information in the texts.

Therefore, providing an automated way to extract information from text data and present it in a structured way will help us reap many benefits and significantly reduce the time we have to spend browsing text documents. This is precisely what information mining strives to achieve. As the sentences are made up of words that belong to different Parts of Speech which normally referred as POS. There are eight different Part of Speech in the English language such as noun, verb, adverb, preposition, conjunction, adjective, intersection and pronoun.

How a specific word functioning in meaning in a given sentence determined by The Parts of Speech (POS). As an example, take the word “good”. In the sentence, “he was concerned with establishing and maintaining his good name”, “good” is used as an adjective. Whereas, in the sentence, “he convinces his father to use his genius for the good of mankind”, “good” is treated as a noun.

This shows that the Parts of Speech tag of a word is very important when we want to understand the meaning of a sentence. And this way can influence to extract meaningful information from our business requirement or functional requirement text.

### 3.4 Algorithmic design



*Figure 3.2 Process flow with NLP And ML Model*

Figure 3.2 shows the process flow with Natural language processing and Machine learning model where several preprocessing stages has involved before proceeding to the ML based information extraction.

Usually, the target information which is related to the requirement text is basically arranged in a few specific patterns. When we look into the Parts of Speech tagging of a certain sentence, they follow with a specific pattern. If we can investigate towards those patterns, we can easily extract the information. Using that extracted information, we can easily create usecase and class diagram for the project purpose.

For the purposes of algorithmic design, we can choose any pos tagger modules for the English Literature. Part of the process of marking parts of speech is to mark the words in a story based on its definition and context for a specific part.

Some POS tagging examples can be mention as: Coordinating Conjunction (CC), Cardinal Digit (CD), existential (EX), adjective (JJ), modal (MD) such as could, will etc., proper noun (NNP), predeterminer (PDT), possessive pronoun (PRP), etc. (Rachiele, G.2018) To assign grammatical information to each word in a sentence we used POS taggers.

Most of the information in a requirement text is made up with verb, gerund/present (VBG), noun plural (NNS), TO, verb base (VB), noun (NN), preposition or subordinating conjunction with these pos taggers. But they come with different patterns.

Sample requirement text would be like below.

The student needs to use the system to register for courses. After the course selection process is completed, the Billing System must be supplied with billing information. The Professor needs to use the system to select the courses to teach for a semester, and must be able to receive a course roster from the system. The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system. User validation must perform when selecting courses to teach and requesting course roster.

Dataset would be like as Table 3.1 shown below according to the above requirement text.

*Table 3.1 Sample dataset*

<b>sentence</b>	<b>actors</b>	<b>use cases</b>	<b>classes</b>
The student needs to use the system to registrar for courses.	Student	Register for courses	Course, Student
After the course selection process is completed, the Billing System must be supplied with billing information to register for courses.	the Billing System	Register for courses	
The Professor needs to use the system to select the courses to teach for a semester, and must be able to receive a course roster from the system.	Professor	Select Courses to teach, Request course roster	Professor, Course
The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Registrar	Creating course catalog, maintain course information, Maintain Professor information, Maintain student information	CourseCatalog Student Professor Registrar
User validation must perform when selecting courses to teach and requesting course roster.	User	Courses to teach, Requesting course roster	User

For now, according to the given sample, there are 3 kinds of specific patterns that can identify:

- {VBG>NNS>TO>VB>NN}
- {NN>IN>DT>NN}
- {VB>DT>NNS>TO>VB}

Like this way we need to experiment more requirement text for find out more feasible patterns to figure out the algorithmic features behind this project. Machine learning techniques we can use to,

- identify actors, usecase and classes
- identify relationship types in use-case diagram and class diagram.

Finding the key terms for class or use case diagram is reflected a classification task in machine learning. The stories narrate different attributes of a diagram and the task is to identify key terms for respective attributes. The use case diagram and class diagram have different set of attributes. The machine learning techniques polarize the data into two clusters; one is training set and the other is testing test. The features from training instances are used for learning purpose of the model. The learned model is later on used for prediction of words for unseen dataset. The comparison of predicted and ground truth key terms assist to estimate performance of the model.

One of the basic statistical techniques depending on the adjustments among the classification result by means of probability and overheads that supplement the conclusion. It follows the theory that problem is probability task and expected values are already available in the space set. The choice is made with instance with minimal error rate. On contrary, where the cost value is not being taken into account for decision making, the class with highest probability is chosen.

### 3.4.1 Preprocess Functional requirement texts

All the words in a document have different chances to appear as a key term in respective attribute. Initially, all the token in a sentence is recognized by means of delimiters. The recurrent words such as propositions don't participate as key terms. However, while calculating the term frequency and inverse document term frequency (TF X IDF), these expressions can dominate other meaningful expressions. Consequently, to deal with these challenging terms we can eradicate from script by allocating probability assessment of zero. Additionally, numerals and non-alphanumeric characters should be removed.

Text pre-processing involves reduction of ambiguities caused by use of several forms of a certain verb, or singular/plural form of a word. Further, stop words, such as a, the, of, is, do not carry much information towards our goal of summarization. Described below are multiple operations employed for preprocessing of documents.

- Document segmentation

A text is divided into several paragraphs to find where each sentence is placed in its respective paragraph.

- Stemming.

We apply stemming to bring a word to its root or base form. The examples include use of a singular form rather than using a plural, or removal of 'ing' from a verb. To this end, Stanford NLP stemmer is employed in this paper. Paragraph segmentation. Paragraph segmentation divides a paragraph into sentences using sentence tags.

- Word normalization.

Each sentence consists of multiple normalized words. Through normalization and lemmatization individual words become one common form, stemming down to their roots. Ambiguities are removed by Porter's algorithm.

- Stop word filtering.

Stop words can be filtered out after performing other preprocessing steps. There is no uniform rule for selecting a stop word because it depends on individual tasks. In this work, words such as a, is, in, the, of are selected as stop words and are filtered out from the document. In text mining applications stop word filtering is considered as a standard step.



## 3.4.2 Machine Learning Techniques to Identify Diagram Elements

### 3.4.2.1 Classification Model

The training data correlated with computed feature value declared formerly, we can take advantage of naïve Bayes classifier in order to identify multiple attributes of diagram. The features for each attribute set are different from other. Hence, we would calculate independent features for all. Based on this assumption we compute separate features and use naïve Bayes for extraction of key terms. Using above-mentioned formula provided with training data, a classification model can be designed and pre-process the data for extracting the key terms such as actors, use case, class and their relationship. Furthermore, other text features like position and distance of text can be integrated to improve accuracy. One of the major hypotheses of naïve Bayes is each feature has independent existence and equally contributes to absolute outcome.

$$\text{Prob(EventA | EventB)} = \text{Prob(EventB | EventA)} \cdot \text{Prob(EventA)} / \text{Prob(EventB)}$$

There are a variety of metrics which can be used to assess how effectively a model works. To determine the optimal evaluation measure for our model, we must first comprehend what every metric evaluates. Although we may hear that a trained model is incredibly accurate, based on the question, what our model is seeking to address, another statistic may be more appropriate for evaluating the model.

### 3.4.2.2 Recurrent neural network Unit

Recurrent neural network (RNN) (Jordan, 1997) is being successfully applied to sequential training issues also including action identification (Donahue et al., 2014), scenery tagging (Byeon et al., 2015), and speech understanding (Cho et al., 2014). A RNN features a recurring connectivity, unlike feed-forward systems like convolution neural network (CNN), how the final hidden stage is an intake towards the forward stage. The following is a description of how stage is updated,

$$h_t = \sigma(Wx_t + Uh_{t-1} + b)$$

The intake and concealed states at  $\tau$  are time interval are  $x_t \in R^M$  and  $h_t \in R^N$ , correspondingly. The values for the present and recurrent inputs, as well as the bias of the neuron, are  $W \in$

$R^{N \times M}$ ,  $U \in R^{N \times N}$ , and  $b \in R^N$ .  $N$  is the count of total neurons in this RNN layers, and  $\sigma$  is a component wise activation function of the neuron. Due to the repetitive manipulation of the recurrence weight matrices, RNN learning struggles out from gradient disappearing and expanding issue. To solve the gradient difficulties, numerous RNN versions have been suggested, including the long short-term memory (LSTM) (Greff et al., 2017; Jozefowicz, Zaremba and Sutskever, 2015) and the gated recurrent unit (GRU) (Cho et al., 2014). As a result, building and learning a broad LSTM or GRU-dependent RNN networks is nearly impossible. Conventional Convolutional Neural Networks with non-soaked activation functions like relu, on the other hand, may be layered into a deep neural network and learned effectively. However, residue interconnections for LSTM systems have been explored in numerous studies (Pradhan and Longpre, 2016; Wu et al., 2016), no substantial enhancement has been achieved. Furthermore, all extant RNN models have the same ingredient as above formula.

$$h_t = \sigma(Wx_t + Uh_{t-1} + b)$$

in which the continuous connections debilitate together all sensory cells. Because the straightforward display of the outcomes of each sensory cell (Karpathy, Johnson and Fei-Fei, 2015) makes it extremely difficult to determine the functionality of one sensory cell without understanding everyone else, it is difficult to comprehend and grasp the functions of the programmed sensory cells (e.g., what sequences each sensory cell reacts to). The autonomously recurrent neural networks (RNNs) is introduced in this study as a current form of RNN. The recurring input parameters are, as

$$s h_t = \sigma(Wx_t + Uh_{t-1} + b)$$

### 3.4.2.3 RNN based Sequence to Sequence Model

This gives a numerous benefit well over standard RNN, which include Machine translation (Bahdanau, Cho and Bengio, 2014; Klein et al., 2017), headlines creation (Ayana et al., 2017; Chopra, Auli and Rush, 2016; Rush, Chopra and Weston, 2015), textual summarization (Chopra, Auli and Rush, 2016), as well as the speech recognition (Donahue et al., 2014) have all been effectively used using sequence to sequence algorithms (Donahue et al., 2014). In the (Rush, Chopra and Weston, 2015), influenced with the achievement of intelligent or neural

machine translations (NMT), developed the neural consideration the sequence to sequence design by the based on attention encoder and the Neural Network Languages Model (NNLM) decoder for extracting key phrase or relation phrase among text, which also outperformed the conventional technique significantly. Improved strategies were proposed (Sutskever, Vinyals and Le, 2014) on this approach by substituting a sequence to sequence RNNs with conventional techniques.

The proposed algorithm includes the recursive encoder and a Recurrent Neural Network and decoding, and it beats some other best models on widely utilized benchmarked datasets, such as in (Chopra, Auli and Rush, 2016) has used a rich-feature the encoder to extract buzzwords. Numerous the text extraction algorithms focused on reducing brief content to phrase descriptions (Sutskever, Vinyals and Le, 2014). The propose method overcomes numerous flaws for content extraction in lengthy materials:

- ✓ They are able to correctly recreate the key elements of primary source materials.
- ✓ They are able to process long sentences effectively.
- ✓ They have a tendency not to repeat words conveying same meaning, resulting in artificial summation.

The proposed recursive RNN based system indirectly blends extracting feature to address the initial two difficulties.

Such style uses a link to extract words from original sentence retain the context. The real data can be correctly duplicated. The effectiveness of these Recursive-RNN based mechanisms has been proven in several following experiments that have attained excellent performance (Jiang and Bansal, 2018).

#### **3.4.2.4 Strategies of training**

Expose the bias and unpredictability of the train and the test measures are two further non-trivial problems by the present the seq2seq system. The Seq2seq architectures are often learned with maximizing the probability of ground-truth words provided by their preceding the ground-truth words and concealed states. Therefore, they are substituted by characters produced by the algorithm. The interpretation errors can rapidly build throughout sequencing formation since the produced symbols or words have never been presented to the decoder throughout learning. Experiential bias is a term used to describe such phenomenon. The discrepancy of readings is another problem.

A curricular training technique called as planned samples is presented to gradually convert the decoder's feed from the ground-truth words to algorithm produced words. As a result, the proposed algorithm overcomes the learning and the testing. It is a feasible method of minimizing susceptibility bias.

### 3.5 Data acquiring methods

In this research mainly based on business requirement or user stories collections, there will be a repository of requirement specification documents, text file contains requirements and documents which contain functional requirements which can be use as training data sets and also as preliminary test-cases with techniques to process them. Data acquiring methods and availability of required data for the research are briefly explained in this section.

For the both modules Data will be acquired through online resources and also from various people and company involve with software developments. Online resources can be various books, white papers and also freely available repositories in world wide web. From this resource category I expect to take texts which contains software description, big picture information about a desired software and specially software requirements or software requirements specification documents. These texts will explain the idea of a software and what will expect to find in terms of requirements. In this module expect to have the feature to identify functional and non-functional requirements most important and valuable resources would be classified requirement text and software requirement specification documents. Some of the resources that can be found in world wide web as follow.

- Software requirement dataset can be find in keggle (<https://www.kaggle.com/iamsouvik/software-requirements-dataset>).
- Labeled requirement dataset can be find in zenodo (<https://zenodo.org/record/268542#.X05hwMgzZhF>).
- Freely available Natural Language Requirements Dataset (<http://fmt.isti.cnr.it/nlreqdataset/>).

these NLP and ML models would be evaluated by adding text which containing the requirements of a software product or project and check result to confirm how far this module able to identify use-case and class diagram elements through the requirement text. This can be

checked easily by referencing the software requirements specification document if available, if not that can be checked manually. Evaluation method for these models with adding only functional requirements and check result to confirm whether the output results are correct. Further we can use, case studies and diagrams drawn for them (as an example BIT Case studies can be used) and also text books having case studies and answers can be used for the evaluation.

This chapter mainly focused on detailed explanation of problem analysis and methodology of the proposed system. First in here describing the problem analysis was performed and the proposed model with design has mentioned thereafter. Then the methodology section started with explaining the methodology that has been used for the research and algorithmic design shown at the next section. In there the sample data sets which is prepared for the model training was shown according to the data set that has been extracted for the project through various sources as mention in the Data acquiring methods section at last. Mainly in this chapter preprocessing techniques for requirement texts and machine learning techniques to identify diagram elements has explained deeply.

# CHAPTER 4

## IMPLEMENTATION

Having discussed the problem analysis and methodology of the proposed system in the previous chapter, in this chapter it will be focused on implementation process of the proposed system with different NLP and ML related techniques and libraries. It starts off with a discussion on the identifying Use case and Class Diagram related elements with NLP and ML techniques for the implementation of the prototype followed by detail explanation of the implementation process and problems encountered with providing proposed solutions upon will also be discussed in using appropriate code snippets

After preparing the initial dataset as Figure 4.1 we have separate it with each component according to the diagram element wise. Then we start to working on model building according to each dataset we have prepared. Python has been used as the programming language and depending on the model and algorithm we chose, relevant library and framework selected for the experiment.

### **4.1 Identifying Use case and Class Diagram Entities with NLP and ML techniques**

After dividing the initial dataset to consists of functional requirement sentences against each diagram element, we have datasets with actors, usecases and classes separately. Thereafter each Data set is divided in to following sets:

Having,

- 10 percent validation pairs,
- 80 percent training pairs
- 10 percent test pairs.

To identify each diagram element from the requirement text, separate machine learning model for each element has built using these training datasets.

In the implementation first preprocessing the data for each model using same the python function as below. Tokenization, finding unique keywords and removing redundant values,

check the words in the english dictionary and removing unused punctuation, removing stopwords etc. are the preprocessing stages done in the implementation.

```
def get_unused_puncs(kws):
    puncs = list(string.punctuation)
    for kw in kws:
        for c in kw:
            if c in puncs:
                puncs.remove(c)
    puncs = re.compile('[ ' + ''.join(puncs) + ' ]')
    return puncs
# Clean tags and titles
train['Tags'] = train['Tags'].str.lower().str.split(' ')
known_kws = np.hstack(train['Tags'].values)
known_kws = np.unique(known_kws)

unused_puncs = get_unused_puncs(known_kws)

stopwords = nltk.corpus.stopwords.words('english')

train['Title'] = train['Title'].str.lower().replace(unused_puncs, ' ').str.split(' ').map(lambda x:
list(set(x).difference(stopwords)))
```

Thereafter extracting feature task was perform where it checks existence of whole keyword and also existence of all parts of known keywords were performed.

```
# 1. Feature: existence of whole keywords

index=0
if type(tag_kws)==float or type(title_kws)==float:
    continue
print(set(title_kws).difference(tag_kws))
for kw in set(title_kws).difference(tag_kws):
    print(index)
    keywords.setdefault(kw, {})
    keywords[kw].setdefault('title', 0)
    keywords[kw]['title'] += 1
print("second")
index=0
for kw in set(title_kws).intersection(tag_kws):

    print(index)
    keywords.setdefault(kw, {})
    keywords[kw].setdefault('both', 0)
    keywords[kw]['both'] += 1
print("Third")
index=0
for kw in set(tag_kws).difference(title_kws):
    print(index)
    keywords.setdefault(kw, {})
    keywords[kw].setdefault('tag', 0)
    keywords[kw]['tag'] += 1
```

```
# 2. Feature: existence of all parts of known keywords
```

```
for kw in tag_kws:  
    if '-' in kw:  
        kw_new = kw.replace('-', ' ')  
        if set(kw_new).issubset(title_kws):  
            keywords[kw].setdefault('both', 0)  
            keywords[kw]['both'] += 1
```

The recurrent words such as propositions don't participate as key terms. Calculating the term frequency and inverse document term frequency (TF X IDF), these expressions can dominate other meaningful expressions. Consequently, to deal with some challenging terms we can eradicate from script by allocating probability assessment of zero and also, numerals and non-alphanumeric characters will be removed.

```
# Calculate scores (posterior prob, tf-idf)
```

```
n = len(train.index)  
removeKeys=[]  
for kw in keywords.keys():  
    keywords[kw].setdefault('title', 0)  
    keywords[kw].setdefault('both', 0)  
    keywords[kw].setdefault('tag', 0)  
    total_tag = keywords[kw]['both'] + keywords[kw]['tag']  
    total_title = keywords[kw]['both'] + keywords[kw]['title']  
    if total_tag == 0:  
        removeKeys.append(kw)  
    elif total_title == 0:  
        removeKeys.append(kw)  
    else:  
        # Posterior probability 'p' = both / (both + title)  
        keywords[kw]['p'] = 1.0 * keywords[kw]['both'] / total_title  
        # tf-idf 'ti' = both * ln( n / (both + title) )  
        keywords[kw]['ti'] = 1.0 * keywords[kw]['both'] * np.log(n / total_title)  
        if keywords[kw]['ti'] <= 1:  
            removeKeys.append(kw)
```

Thereafter naïve based classifier has been implemented and as previous stages clean and tokenization tasks also perform before generate predicted tags based on decision rule.

```
def nb_classify(test, keywords):
```

```
    unused_puncs = get_unused_puncs(keywords.keys())  
    stopwords = nltk.corpus.stopwords.words('english')  
    test['Title'] = test['Title'].str.lower().replace(unused_puncs, ' ').str.split(' ').map(lambda  
x: list(set(x).difference(stopwords)))  
  
    pred = test[['Id']]  
    pred['Tags'] = pd.Series(test['Title'].map(lambda x: decision_rule(x, keywords,  
stopwords)).map(lambda x: ' '.join(x)))  
  
    return pred
```



In the decision rule method, it takes the functional text, and the keywords from the early stage perform and also stopwords. First it gets scores of each whole keywords from the functional text provided. Then get scores for each known keywords with all parts in the requirement sentence. After that, three steps performed as,

1. adding tag if posterior probability is higher than or equal to 0.5
2. add highest scoring to TF-IDF keywords if not already added.
3. Add 'c#' if there no tags ('c#' just a special keyword to identify it clearly from other keywords)

```
def decision_rule(title_kws, keywords, stopwords):

    # Get scores of each whole kw in title
    nb_scores = {}
    ti_scores = {}
    for kw in set(title_kws).intersection(keywords.keys()):
        nb_scores[kw] = keywords[kw]['p']
        ti_scores[kw] = keywords[kw]['ti']

    # Get scores for each known kw with all parts in title
    for kw in filter(lambda x: '-' in x, keywords.keys()):
        kw_new = set(kw.replace('-', ' ').difference(stopwords))
        if kw_new.issubset(title_kws):
            nb_scores[kw] = keywords[kw]['p']
            ti_scores[kw] = keywords[kw]['ti']

    # 1
    kws_to_tag = []
    for kw in sorted(nb_scores, key=nb_scores.get, reverse=True):
        if nb_scores[kw] >= 0.5:
            kws_to_tag.append(kw)

    # 2
    for kw in sorted(ti_scores, key=ti_scores.get, reverse=True):
        if kw not in kws_to_tag:
            kws_to_tag.append(kw)
            break

    # 3
    if len(kws_to_tag) == 0:
        kws_to_tag.append('unknown')

    return kws_to_tag
```

Following above procedures separate models created to identify actors, usecase and classes from the functional requirement text. Those models trained using training dataset and evaluation performed using test dataset. We have to perform model training for each actor, usecase and class Models and store those in a file location to predict using different dataset (validation dataset) in the evaluation stage.

## 4.2 Identifying Use case and Class Diagram Relationships with NLP and ML techniques

In this implementation our focus mainly on identifying relationship in both use case and class diagram related components separately. This is a combination of two models where one model will be identifying relationship involved entities. As an example of use case diagram first model will identify what are the actors and use cases involve in the relationship. Second model will be identifying the relationship type of that particular relationship. These two models need to build for both use case and class diagram separately. Two models per each diagram type to identify relationship involved in particular functional requirement text.

As previous implementation, in this implementation also we dividing the initial dataset to consists of functional requirement sentences against each diagram element which we need this time usecase relationship and class relationship. Each dataset, again we need to separate for the two models as explained above paragraph, where one dataset with relationship involved entities and other dataset with relationship type along with the functional text of each. Each data set is divided in to following sets:

Having,

- 10 percent validation pairs,
- 80 percent training pairs
- 10 percent test pairs.

To identify each diagram relationship from the requirement text, two separate machine learning models for each diagram type has built using these training datasets.

Below mentioning the implementation of model which identifying the diagram entities involving in the Relationship. For this RNN based Sequence to Sequence Model applied train and build this model. 'Keras' framework and its related libraries used for this implementation and important code snippets shown below from the implementation.

```

def transform_input_text(self, texts):
    temp = []
    for line in texts:
        x = []
        for word in line.lower().split(' '):
            wid = 1
            if word in self.input_word2idx:
                wid = self.input_word2idx[word]
            x.append(wid)
            if len(x) >= self.max_input_seq_length:
                break
        temp.append(x)
    temp = pad_sequences(temp, maxlen=self.max_input_seq_length)

    print(temp.shape)
    return temp

```

In above code input text will transform to a sequence of text and below code will split the target text.

```

def split_target_text(self, texts):
    temp = []
    for line in texts:
        x = []
        line2 = 'START ' + line.lower() + ' END'
        for word in line2.split(' '):
            x.append(word)
            if len(x)+1 >= self.max_target_seq_length:
                x.append('END')
                break
        temp.append(x)
    return temp

```

```

def generate_batch(self, x_samples, y_samples, batch_size):
    encoder_input_data_batch = []
    decoder_input_data_batch = []
    decoder_target_data_batch = []
    line_idx = 0
    while True:
        for recordIdx in range(0, len(x_samples)):
            target_words = y_samples[recordIdx]
            x = x_samples[recordIdx]
            decoder_input_line = []

            for idx in range(0, len(target_words)-1):
                w2idx = 0
                w = target_words[idx]
                if w in self.target_word2idx:
                    w2idx = self.target_word2idx[w]
                decoder_input_line = decoder_input_line + [w2idx]
                decoder_target_label = np.zeros(self.num_target_tokens)
                w2idx_next = 0
                if target_words[idx+1] in self.target_word2idx:
                    w2idx_next = self.target_word2idx[target_words[idx+1]]
                if w2idx_next != 0:
                    decoder_target_label[w2idx_next] = 1

            decoder_input_data_batch.append(decoder_input_line)
            encoder_input_data_batch.append(x)
            decoder_target_data_batch.append(decoder_target_label)

            line_idx += 1
            if line_idx >= batch_size:
                yield [pad_sequences(encoder_input_data_batch, self.max_input_seq_length),
                       pad_sequences(decoder_input_data_batch,
                                     min(self.num_target_tokens,
                                         RecursiveRNN2.MAX_DECODER_SEQ_LENGTH))], np.array(decoder_target_data_batch)
                line_idx = 0
                encoder_input_data_batch = []
                decoder_input_data_batch = []
                decoder_target_data_batch = []

```

Above function will generate batch with taking input and output text.

After identifying diagram entities involving in the relationship, second model will identify the relationship type those entities involved with. Important code snippets shown below from that model implementation.

In below code showing the Convolution Neural network classifier for text classification to identify relationship type.

```

for i, filter_size in enumerate(self.filter_sizes):
    with tf.variable_scope("conv-maxpool-%s" % filter_size):
        # Convolution
        filter_shape = [filter_size, self.embedding_size, 1, self.num_filters]
        W = tf.get_variable("weights", filter_shape,
initializer=tf.truncated_normal_initializer(stddev=0.1))
        b = tf.get_variable("biases", [self.num_filters], initializer=tf.constant_initializer(0.0))

        conv = tf.nn.conv2d(inputs,
                            W,
                            strides=[1, 1, 1, 1],
                            padding='VALID',
                            name='conv')
        # Activation function
        h = tf.nn.relu(tf.nn.bias_add(conv, b), name='relu')

        # Maxpool
        pooled = tf.nn.max_pool(h,
                                ksize=[1, self.max_length - filter_size + 1, 1, 1],
                                strides=[1, 1, 1, 1],
                                padding='VALID',
                                name='pool')
        pooled_outputs.append(pooled)

```

In below code it adding regularization to output layer and calculate loss and accuracy of each iteration of input.

```

with tf.name_scope('softmax'):
    softmax_w = tf.Variable(tf.truncated_normal([num_filters_total, self.num_classes], stddev=0.1),
name='softmax_w')
    softmax_b = tf.Variable(tf.constant(0.1, shape=[self.num_classes]), name='softmax_b')

# Add L2 regularization to output layer
self.l2_loss += tf.nn.l2_loss(softmax_w)
self.l2_loss += tf.nn.l2_loss(softmax_b)

self.logits = tf.matmul(h_drop, softmax_w) + softmax_b
predictions = tf.nn.softmax(self.logits)
self.predictions = tf.argmax(predictions, 1, name='predictions')

# Loss
with tf.name_scope('loss'):
    losses = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=self.input_y, logits=self.logits)
    # Add L2 losses
    self.cost = tf.reduce_mean(losses) + self.l2_reg_lambda * self.l2_loss

# Accuracy
with tf.name_scope('accuracy'):
    correct_predictions = tf.equal(self.predictions, self.input_y)
    self.correct_num = tf.reduce_sum(tf.cast(correct_predictions, tf.float32))
    self.accuracy = tf.reduce_mean(tf.cast(correct_predictions, tf.float32), name='accuracy')

```

The proposed model is trained on CPU system with 32GB RAM. In our experiments, dimension of hidden states and word embeddings are 256 and 128 respectively. For source and target vocabulary of roughly 22, 00 words is used for training the network. Proposed network is capable of retaining the context of words; hence we can employ shorter vocabulary size. Proposed recursive RNN structure introduces very low number of learnable parameters to network. Recursive property adds 153 and spare parameters in network. Contrary to train embedding from scratch for smaller number of words we have employed pre-trained embedding for words representation. We have trained our network with learning rate of 0.0001. Accumulator value is set to 0.1 using Adagrad. For regularization, we have employed early stopping. We concurrently run our model in training and evaluation mode. Early stopping ceases network training when it seems over fit.

### 4.3 Draw diagram with identified diagram elements

With this implementation we want to generate the usecase and class diagram according to identified elements through previous two implementation. Usecase and class diagram elements that has been identified from functional requirement text has outputted as JSON file format. Sample JSON output structure can be seen as Figure 4.1.

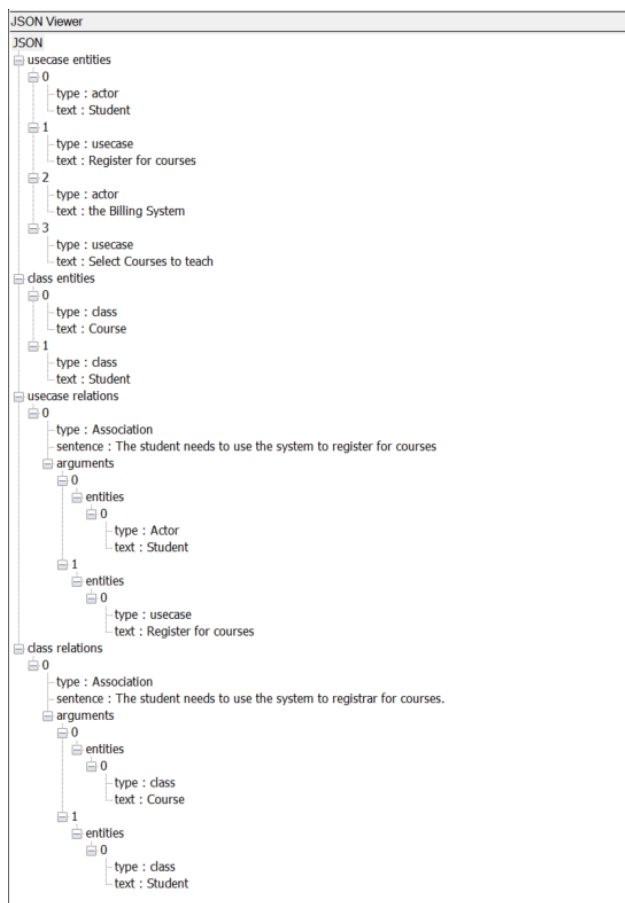


Figure 4.1 Sample JSON output format

According to the generated JSON format generate the plantUML markup to render was the goal of this implementation. After generating plantUML markup text, with using 'plantuml' python library where we can simply generate the respective diagrams. Implementation shown below.

```
def generateUsecaseDiagram(usecaseEntities,usecaseRelations):
    useCases, actors, relations = set(), set(), set()

    #get all usecase diagram entities
    for usecaseEntity in usecaseEntities:
        if(usecaseEntity.get("type")) and (usecaseEntity.get("type") == "actor"):
            actor = "".join([actor.strip().capitalize() for actor in usecaseEntity.get("text").split(" ")])
            actors.add("actor " + actor)
        elif(usecaseEntity.get("type")) and (usecaseEntity.get("type") == "usecase"):
            useCases.add("usecase (" + usecaseEntity.get("text").strip().capitalize() + ")")

    #get all usecase diagram relationships
    for usecaseRelation in usecaseRelations:
        if(usecaseRelation.get("type")) and (usecaseRelation.get("type") == "Association"):
            relations.add(drawTextToUsecaseRelation("Association",usecaseRelation.get("arguments"),useCases, actors))
        elif(usecaseRelation.get("type")) and (usecaseRelation.get("type") == "Include"):
            relations.add(drawTextToUsecaseRelation("Include",usecaseRelation.get("arguments"),useCases, actors))
        elif(usecaseRelation.get("type")) and (usecaseRelation.get("type") == "Extends"):
            relations.add(drawTextToUsecaseRelation("Extends",usecaseRelation.get("arguments"),useCases, actors))
        elif(usecaseRelation.get("type")) and (usecaseRelation.get("type") == "Generalization"):
            relations.add(drawTextToUsecaseRelation("Extends",usecaseRelation.get("arguments"),useCases, actors))

    #generate The plantUML markup to render
    plantuml_text = generateUsecaseDiagramTxt(useCases, actors, relations)

    #Return the server URL for the image
    usecaseDiagramURL = plantumlObj.get_url(plantuml_text)

    return usecaseDiagramURL,plantuml_text
```

```

def generateClassDiagram(classEntities,classRelations):
    clazzes, relations = set(), set()

    #get all class diagram entities
    for classEntity in classEntities:
        if(classEntity.get("type")) and (classEntity.get("type") == "class"):
            clazz = "".join([clazz.strip().capitalize() for clazz in classEntity.get("text").split(" ")])
            clazzes.add("class " + clazz)

    #get all class diagram relationships
    for classRelation in classRelations:
        if(classRelation.get("type")) and (classRelation.get("type") == "Association"):
            relations.add(drawTextToClassRelation("Association",classRelation.get("arguments"),clazzes))
        elif(classRelation.get("type")) and (classRelation.get("type") == "Aggregation"):
            relations.add(drawTextToClassRelation("Aggregation",classRelation.get("arguments"),clazzes))
        elif(classRelation.get("type")) and (classRelation.get("type") == "Composition"):
            relations.add(drawTextToClassRelation("Composition",classRelation.get("arguments"),clazzes))
        elif(classRelation.get("type")) and (classRelation.get("type") == "Dependency"):
            relations.add(drawTextToClassRelation("Dependency",classRelation.get("arguments"),clazzes))
        elif(classRelation.get("type")) and (classRelation.get("type") == "Generalization"):
            relations.add(drawTextToClassRelation("Generalization",classRelation.get("arguments"),clazzes))

    #generate The plantUML markup to render
    plantuml_text = generateClassDiagramTxt(clazzes, relations)

    #Return the server URL for the image
    classDiagramURL = plantumlObj.get_url(plantuml_text)

    return classDiagramURL,plantuml_text

```

This chapter discussed about the implementation process of the proposed UML Diagrams Generator system. Chapter started with justifying the use of various ML models for the identifying different diagram elements on the both usecase and class diagram. Mainly in identifying use case and class diagram entities with NLP and ML techniques describe as the first section and after identifying use case and class diagram relationships with NLP and ML techniques described. After wards implementation for the drawing diagram with identified diagram elements was described. each section in this chapter was describe with the explanation through code snippets for better understanding of the implementation.



# **CHAPTER 5**

## **EVALUATION AND RESULTS**

In this chapter the results of the implementation are presented and discussed evaluation with reference to the aim of the study, which was to generating UML diagrams against Business Requirements Using Natural language processing and Machine learning. The two sub-aims - the first to find the most appropriate NLP and ML techniques towards identifying use case and class diagrams elements from business requirement, and the second to generate diagram using the diagram generating tool form the main comparisons in the evaluation. Evaluation was done by using a case study and solution given in a text book with the project output.

To describe the functional requirements, we used UML use case diagram and class diagrams as two different approaches. Natural Language Processing (NLP) and Machine learning (ML) techniques would use to extract the information because the requirements are written in natural language. The aim of this project is to setup an approach to automate in generating the UML use case diagrams and class diagrams from the business requirements text using natural language processing and Machine learning technologies.

### **5.1 Collecting Dataset**

This proposed system will be adhering to experimental research methodology. Proposed system will contain with two main approaches to be considered as identify use case elements through business requirement and identify class Diagram elements through business requirement. Each approach evaluation data availability briefly explains as follows.

For the both NLP and ML models required Data will be acquired through online resources and also from various people and company involve with software developments. Online resources can be various books, white papers and also freely available repositories in world wide web. From this resource category, expect to take texts which contains software description, big picture information about a desired software and specially software requirements or requirement text from software requirements specification documents. These texts will explain the idea of a software and what will expect to find in terms of requirements. In this module expect to have the feature to identify use case and class diagram required elements most important and valuable resources would be classified requirement text and software requirement specification documents. Some of the resources that can be found in world wide web as follow.

- Software requirement dataset can be found in Kaggle and just needed functional requirements. (<https://www.kaggle.com/iamsouvik/software-requirements-dataset>)
- Labeled requirement dataset can be find in Zenodo and need to extract functional requirements from them. (<https://zenodo.org/record/268542#.X05hwMgzZhF>)
- Freely available Natural Language Requirements Dataset in National Research council of Italy with software Requirement specification documents, from them functional requirements needed to extract. (<http://fmt.isti.cnr.it/nlreqdataset/>)

## 5.2 Dataset Preparation

Dataset has been created with functional requirements which has been extracted through above mentioned sources. Sample dataset format model training created as below Figure 5.1.

	A	B	C	D	E	F	G
		actors	usecases	usecase relationship	classes	class relationship	Class attributes
1	The student needs to use the system to registrar for courses.	Student	Register for courses	Association(Student --- Register for courses)	Course, Student	Association(Course -- Student)	
2	After the course selection process is completed, the Billing System must be supplied with billing information to register for courses.	the Billing System	Register for courses	Association(the Billing System --- Register for courses)			
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Professor	Select Courses to teach, Request course roster	Association(Professor --- Select Courses to teach), Association(Professor --- Request course roster)	Professor, Course	Association(Course -- Student)	
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Registrar	Creating course catalog, Maintain course information, Maintain Professor information, Maintain student information	Association(Registrar --- Creating course catalog), Association(Registrar --- Maintain course information), Association(Registrar --- Maintain Professor information), Association(Registrar --- Maintain student information)	CourseCatalog, Student, Professor, Registrar		
5	User validation must perform when selecting courses to teach and requesting course roster.		select Course to teach, Request course roster, validate user	Include(validate user <- select Course to teach ), Include(validate user <- Request course roster )	User, Course		
6	The system shall display Events or Activities.	User	display event, display activity				
7	The display shall have two regions: left 2/3 of the display is graphical right 1/3 of the display is a data table				region, Graphical region, data table	Generalization(region <- Graphical region), Generalization(region <- data table)	
8	The data displayed in both the nodes within the graph and the rows in the table are MSEL Summary data		display data		MSEL Summary data		
9	The table side of the display shall be split into 2 regions: sequential and temporal.				region, sequential region, temporal region	Generalization(region <- sequential region), Generalization(region <- temporal region)	
10	The top 1/4 of the table will hold events that are to occur sequentially.		hold events, sequentially occur events	Generalization(hold events<-sequentially occur events)	Event, sequentially occur event	Generalization(event<-sequentially occur event)	
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.		hold events, current time occur events	Generalization(hold events<-current time occur events)	Event, current time event	Generalization(event<-current time occur event)	
12	Program Administrators and Nursing Staff Members shall be able to add a new cohort to the system identified by start month and year.	Program Administrator, Nursing Staff Member	add new cohort	Association (Program Administrator/ Nursing Staff Member -- add new cohort)	cohort, Program Administrator, Nursing Staff Member		cohort(startmonth, start year)
13	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	Program Administrator, Nursing Staff Member	add new class	Association (Program Administrator/ Nursing Staff Member -- add new class)	department, class, Program Administrator, Nursing Staff Member	Association(department -- class)	
14	A class shall be either a non-clinical class or a clinical class.				class, clinical class, non-clinical class	Generalization(Class<-clinical class), Generalization(Class<-non clinical class)	
15	A non-clinical class shall specify the course name lecture room requirements and instructor needs.				non-clinical class		non-clinical class(course name, lecture room requirements, instructor
16	A clinical class shall specify the course name lecture room requirements clinical site needs lecture instructor needs and clinical lab instructor needs.				clinical class		clinical class (course name, lecture room requirements, clinical site needs, lecture instructor needs, clinical lab instructor needs)
17							
18							
19							

Figure 5.1 Sample dataset format with functional requirement

Both these NLP and ML models will be evaluated by adding text containing the requirements of a software product or project and check result to confirm how far this module able to identify use-case and class diagram elements through the requirement text. Also, it's worth to evaluate using a Case study and Solution given in a text book with the model output.

### 5.3 Evaluation approach

Accuracy of identifying elements to draw use case and class diagram can be checked easily by referencing Case study and Solution given in a text book. Also referring the software requirements specification document diagrams if those available, if not then that can be checked by manually identifying. Then we can check result to confirm whether the output results are same or different. From that we can be able to evaluate that, model was able to identify relevant elements which are useful to draw complete use case and class diagram.

To evaluate the Model, we can use the term called Model Evaluation Metrics.

Model evaluation metrics are needed to measure model performance. The choice of evaluation parameters depends on the machine learning task provided. In this project that is classification.

#### Classification metrics

When performing predictions through classification, four kinds of result that could occur as follow.

- **True positives** are when we assume an observation belongs to a particular class and actually it is belonging to that exact class.
- **True negatives** are when we assume an observation is not belonging to a particular class and that actually not belonging to that particular class as expected.
- **False positives** occur when we predict our observation belongs to a one particular class but in reality, it is not belonging to that class.
- **False negatives** occur when we assume an observation is not belonging to a particular class but actually it is belonging to that same class.

Above mentioned four types of results are usually plotted on a confusion matrix. Following confusion matrix in Figure 5.2 can be presented as an example for the case of binary classification.

		<b>Prediction</b>	
		1	0
<b>True Label</b>	1	True negatives	False positives
	0	False negatives	True positives

Figure 5.2 confusion matrix for the case of binary classification

Accuracy, precision, and recall are the three main metrics that can be used to evaluate a classification model in generally.

**Accuracy** – In here, this is defining as the percentage of correctly predicted predictions of the test data. This can be calculated simply with dividing the number of correctly predicted predictions by the total number of all predictions.

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{All Predictions}}$$

**Precision** – This would be defined as a fraction of true positives which means relevant examples among all of the examples as this will predicted to belong in a particular class. Simply this means how many relevant items were predicted.

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}$$

**Recall** – This is also defined as the fraction of test data which were predicted as belonging to a particular class with respect to all of the test data that truly belong in the class. Simply it means how many relevant items are predicted.

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negative}}$$

To evaluate both the precision and recall of a model is important. Therefore, it's better to have one number to evaluate a machine learning model with including this precision and recall metrics. Thus, it makes sense to have a common approach for combining these metrics. f-score is introduced to address that concern.

$$F_{\beta} = (1+\beta^2) \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

In here  $\beta$  parameter allows to control the tradeoff value as it is important between precision and recall. When  $\beta < 1$  means focuses more on precision, and  $\beta > 1$  indicating focuses more on recall. In this case, it's better to focus more on the model's recall than its precision.

## 5.4 Evaluation and Results

Detailed discussions of the evaluation results shown below.

### 5.4.1 Identifying use case diagram element from requirement text

Machine Learning model has able to identified use case diagram element such as actors, use-cases to acceptable extend since the three main metrics Precision, Recall and F1 score showed good scores as below.

#### Identifying Actors

After execution of the program against the functional requirement text identified actors by ML model and test dataset used to evaluate shown in below Table 5.1, Table 5.2 and Table 5.3.

Table 5.1 Test data snippet for actor in functional text

<b>Id</b>	<b>Functional Text</b>	<b>Actual actors</b>
1	The student needs to use the system to registrar for courses.	student
2	After the course selection process is completed, the Billing System must be supplied with billing information to register for courses.	the billing system
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	professor
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	registrar
5	User validation must perform when selecting courses to teach and requesting course roster.	user
17	Realtor shall be able to generate a CMA report based on property criteria	realtor
53	Citizens can register their complaints with police and then based on the evidence, facts and following investigation, police shall take the complaint forward.	citizen, police
54	The Registration module acts as an interface between the police and citizens and it eases the approach, interaction and information exchange between police and complainants.	citizen, police
55	After a complaint is initiated, police initiates the investigation process.	police
58	A designated constable from each police station constantly interfaces with the courts.	police station, constable
60	The Search module of the CCTNS gives police personnel the ability to execute a basic or advanced search on cases.	police
61	Using the search functionality, police personnel can search for a particular person, type of crime, modus operandi, property etc.	police
62	search module also gives the police personnel the ability to customize the results view by criminal/accused or by cases.	police
63	It makes reporting easy for police by enabling them to execute different types of queries such as monthly reporting, RTI related etc.	police
69	system shows information such as cases assigned, alerts, pending tasks etc hence helping police personnel to plan better and execute with greater efficiency.	police
71	With a proper configuration, information such as act and sections, state specific data, castes, tribes, property information etc. can be created updated and deleted by police personnel.	police

Table 5.2 Model Predicted results snippet for identified actors

<b>Id</b>	<b>Model Identified actors</b>
1	student
2	the billing system
3	professor
4	registrar
5	user
17	realtor
53	-
54	citizen, police
55	police
58	-
60	police
61	police
62	police
63	police
69	police
71	police

The evaluation result of the three main metrics used to evaluate a classification model as below.

Table 5.3 Evaluation metric result for actors

Evaluation metric	Score
Mean Precision	0.43504
Mean Recall	0.78504
Mean F1 score	0.53701

## Identifying Use case

Identifying use cases from the requirement text also had good evaluation metric scores. Test and it's predicted data set can find on below Table 5.4 and 5.5.

Table 5.4 Test data snippet for usecase in functional text

Id	Functional text	Actual usecases
1	The student needs to use the system to registrar for courses.	register for courses
2	After the course selection process is completed, the Billing System must be supplied with billing information to register for courses.	register for courses
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	select courses to teach, request course roster
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	creating course catalog, maintain course information, maintain professor information, maintain student information
5	User validation must perform when selecting courses to teach and requesting course roster.	select course to teach, request course roster, validate user
6	The system shall display Events or Activities.	display event, display activity
8	The data displayed in both the nodes within the graph and the rows in the table are MSEL Summary data	display data
12	The system shall color code events according to their variance from current time.	color code events
13	The system shall provide a history report of changes made to the Activity or Event data	provide a history report
16	The system shall update or create new property listings in the MLS	create new property, update property
17	Realtor shall be able to generate a CMA report based on property criteria	generate cma report
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	add clinical class or section
19	Program Administrators and Nursing Staff Members shall be able to add a new cohort to the system identified by start month and year.	add new cohort
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	add new class
24	Program Administrators/Nursing Staff Members shall be able to create a new Program of Study.	create new program
26	Program Administrators and Nursing Staff Members shall have the ability to specify which classes are required for a Program of Study	specify required classes for program



Table 5.5 Model Predicted results snippet for identified usecases

<b>Id</b>	<b>Model Identified usecases</b>
1	student courses
2	register course courses information billing
3	select course courses roster request
4	course information
5	course courses roster validation
6	display events
8	summary graph data table
12	code events color time
13	history report changes data provide
16	update new create property
17	generate cma report property
18	add clinical classes
19	add new cohort
20	add new classes
24	new create nursing
26	specify required classes

Evaluation metrics calculation for identifying usecase from functional requirement text as below Table 5.6.

Table 5.6 Evaluation metric result for usecases

<b>Evaluation metric</b>	<b>Score</b>
Mean Precision	0.69624
Mean Recall	0.62698
Mean F1 score	0.62788

### **Identifying use case relationships**

Identifying use case relationships was done with two models as explained in Implementation chapter. First model identifying relationship involved entities. After several training with parameter changing, it was able to generate good result.

In Table 5.7 shows the data snippet from the preprocessing stage data separation to identify use case relationship according to involved entity types. As an example, whether that functional text having actor to use case relationship or use case to use case relationship. Table 5.8 shows the Test data snippet used to Test the first ML model and Table 5.9 shows the predicted result.

Table 5.7 Data snippet from preprocessing stage data separation

Id	Functional Text	actor to usecase	usecase to usecase
1	The student needs to use the system to registrar for courses.	Student --- Register for courses	
2	After the course selection process is completed, the Billing System must be supplied with billing information to register for courses.	the Billing System --- Register for courses	
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Professor --- Select Courses to teach Professor --- Request course roster	
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Registrar --- Creating course catalog Registrar --- Maintain course information Registrar --- Maintain Professor information Registrar --- Maintain student information	
5	User validation must perform when selecting courses to teach and requesting course roster.		validate user <-- select Course to teach validate user <-- Request course roster
10	The top 1/4 of the table will hold events that are to occur sequentially.		event<-sequentially occur event
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.		event<-current time occur event
17	Realtor shall be able to generate a CMA report based on property criteria	Realtor --- generate CMA report	
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Program Administrator/ Nursing Staff Member -- add clinical class or section	
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Program Administrator/ Nursing Staff Member -- add clinical class or section	
19	Program Administrators and Nursing Staff Members shall be able to add a new cohort to the system identified by start month and year.	Program Administrator/ Nursing Staff Member -- add new cohort	
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	Program Administrator/ Nursing Staff Member -- add new clasS	

Table 5.8 Test data snippet for use case relationship involved entities in functional text

id	Functional Text	title
1	The student needs to use the system to register for courses.	Student Register for courses
2	After the course selection process is completed, the Billing System must be supplied with billing information to register for courses.	the Billing System Register for courses
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Professor Select Courses to teach
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Professor Request course roster
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Registrar Creating course catalog
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Registrar Maintain course information
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Registrar Maintain student information
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Registrar Maintain Professor information
5	User validation must perform when selecting courses to teach and requesting course roster.	validate user
5	User validation must perform when selecting courses to teach and requesting course roster.	Request course roster
10	The top 1/4 of the table will hold events that are to occur sequentially.	Event sequentially occur event
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.	Event current time occur event
17	Realtor shall be able to generate a CMA report based on property criteria	Realtor generate CMA report
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Program Nursing Staff Member add clinical class or section
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Program Administratorf Member add clinical class or section
19	Program Administrators and Nursing Staff Members shall be able to add a new cohort to the system identified by start month and year.	Program Administrator add new cohort
19	Program Administrators and Nursing Staff Members shall be able to add a new cohort to the system identified by start month and year.	Nursing Staff Member add new cohort
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	Program Administrator add new clasS
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	Nursing Staff Member add new clasS

Table 5.9 Model Predicted results snippet for identified use case relationship involved entities

Id	Functional Text	title
1	The student needs to use the system to register for courses.	student register for courses
2	After the course selection process is completed, the Billing System must be supplied with billing information to register for courses.	the billing system register for courses
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	professor select courses to teach
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	registrar maintain student information
5	User validation must perform when selecting courses to teach and requesting course roster.	request course roster
10	The top 1/4 of the table will hold events that are to occur sequentially.	event sequentially occur event
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.	event current time occur event
17	Realtor shall be able to generate a CMA report based on property criteria	realtor generate cma report
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	program nursing staff member add clinical class or section
19	Program Administrators and Nursing Staff Members shall be able to add a new cohort to the system identified by start month and year.	program administrator add new cohort
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	nursing staff member add new class

Second model will be identifying the relationship type of that particular relationship that can identify in respective functional requirement text. Table 5.10 shows the Test data snippet used to test the second ML model and Table 5.11 shows the predicted result of it.

Table 5.10 Test data snippet for use case relationship Type in functional text

<b>Id</b>	<b>Functional Text</b>	<b>Relationship Type</b>
1	The student needs to use the system to registrar for courses.	Association
2	After the course selection process is completed, the Billing System must be supplied with billing information to register for courses.	Association
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Association
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Association
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Association
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Association
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Association
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Association
5	User validation must perform when selecting courses to teach and requesting course roster.	Include
5	User validation must perform when selecting courses to teach and requesting course roster.	include
10	The top 1/4 of the table will hold events that are to occur sequentially.	Generalization
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.	Generalization
17	Realtor shall be able to generate a CMA report based on property criteria	Association
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Association
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Association
19	Program Administrators and Nursing Staff Members shall be able to add a new cohort to the system identified by start month and year.	Generalization
19	Program Administrators and Nursing Staff Members shall be able to add a new cohort to the system identified by start month and year.	Generalization
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	Generalization
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	Generalization

Table 5.11 Model Predicted results snippet for identified use case relationship type

<b>Id</b>	<b>Functional Text</b>	<b>Relationship Type</b>
1	The student needs to use the system to register for courses.	Association
2	After the course selection process is completed, the Billing System must be supplied with billing information to register for courses.	Association
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Association
4	The Registrar is responsible for the generation of the course catalog for a semester, and for the maintenance of all information about the curriculum, the students, and the Professors needed by the system.	Association
5	User validation must perform when selecting courses to teach and requesting course roster.	Association
10	The top 1/4 of the table will hold events that are to occur sequentially.	Association
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.	Association
17	Realtor shall be able to generate a CMA report based on property criteria	Association
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Association
19	Program Administrators and Nursing Staff Members shall be able to add a new cohort to the system identified by start month and year.	Association
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	Association

## 5.4.2 Identifying class diagram element from requirement text

Trained NLP and ML model able to identify class diagram elements from the functional requirement input text. This project is only considered on class diagram elements such as classes and identifying classes that have relationship. Due to limited amount of data relationship type identifying is difficult with this phase. Identifying classes within the functional requirement text has acceptable result as the three main metrics Precision, Recall and F1 score showed good scores as below.

### Identifying Classes

Identifying classes from the requirement text had good evaluation metric scores. Test dataset and it's predicted data set can find on below Table 5.12 and Table 5.13 respectively.

Table 5.12 Test data snippet for classes in functional text

Id	Functional text	Actual classes
1	The student needs to use the system to registrar for courses.	course, student
5	User validation must perform when selecting courses to teach and requesting course roster.	user, course
8	The data displayed in both the nodes within the graph and the rows in the table are MSEL Summary data	mset summary data
22	A non-clinical class shall specify the course name lecture room requirements and instructor needs.	non-clinical class
23	A clinical class shall specify the course name lecture room requirements clinical site needs lecture instructor needs and clinical lab instructor needs.	clinical class
24	Program Administrators/Nursing Staff Members shall be able to create a new Program of Study.	program, program administrator, nursing staff member
25	A Program of Study shall consist of a program name and listing of required classes (both clinical and non-clinical) that must be completed.	program
26	Program Administrators and Nursing Staff Members shall have the ability to specify which classes are required for a Program of Study	program administrator, nursing staff member, program, class
27	The system shall be able to display a report of needed classes for a given quarter for all cohorts of all programs for Program Administrators/Nursing Staff Members planning purposes.	report
28	The report of needed classes shall include (but not be limited to) classes to be offered number of sections needed number of labs needed and room types needed.	report
30	Program Administrators and Nursing Staff Members shall be able to add a new clinical site into the system.	clinical site, program administrator, nursing staff member
31	Clinical site information shall include (but not be limited to) the name of the site contact person and contact information.	clinical site
106	The leads washing functionality will store any potential lead duplicates returned by the enterprise system.	lead, enterprise
127	At the start of each turn the product shall notify each player of his or her status.	player
138	If the shot was marked as a miss the product shall change the status of the offensive player to defensive	player
139	If the shot was marked as a miss the product shall change the status of the defensive player to offensive	player

Table 5.13 Model Predicted results snippet for identified classes

<b>Id</b>	<b>Model Identified classes</b>
1	student
5	course
8	mset summary data
22	non-clinical class
23	clinical lab class
24	staff nursing program
25	non-clinical clinical program
26	staff nursing program
27	staff nursing program report
28	report
30	staff nursing clinical program site
31	clinical site
106	lead enterprise
127	player
138	player offensive defensive shot
139	player offensive defensive shot

Evaluation metrics calculation for identifying classes from functional requirement text as below Table 5.14.

Table 5.14 Evaluation metric result for classes

<b>Evaluation metric</b>	<b>Score</b>
Mean Precision	0.80752
Mean Recall	0.71011
Mean F1 score	0.71504

### Identifying Class Relationships

Same as usecase relationships, identifying class relationship also done with two models as explained in Implementation chapter. First model identifying relationship involved entities. After several training with parameter changing, it was able to generate good result.

In Table 5.15 shows the data snippet from the preprocessing stage data separation to identify class relationship according to involved entity types. Table 5.16 shows the Test data snippet used to Test the first ML model and Table 5.17 shows the predicted result.

Table 5.15 Data snippet from preprocessing stage class relationship data

Id	Functional Text	Class Relationship
1	The student needs to use the system to registrar for courses.	Association(Course -- Student)
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Association(Professor -- Course)
7	The display shall have two regions: left 2/3 of the display is graphical right 1/3 of the display is a data table	Generalization(region <- Graphical region), Generalization(region <- data table)
9	The table side of the display shall be split into 2 regions: sequential and temporal.	Generalization(region <- sequential region), Generalization(region <- temporal region)
10	The top 1/4 of the table will hold events that are to occur sequentially.	Generalization(event<-sequentially occur event)
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.	Generalization(event<-current time occur event)
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Generalization(Class<-clinical class)
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	Association(department -- class)
21	A class shall be either a non-clinical class or a clinical class.	Generalization(Class<-clinical class), Generalization(Class<-non clinical class)
26	Program Administrators and Nursing Staff Members shall have the ability to specify which classes are required for a Program of Study	Association(Program -- Class )
32	Program Administrators and Nursing Staff Members shall be able to add a new clinical lab section for an existing clinical class into the System.	Association (Clinical class -- clinical lab section)
34	Program Administrators and Nursing Staff Members shall be able to add a student who has registered for a clinical class to a clinical lab section for that class.	Association (Student -- Clinical class), Association(Clinical class -- clinical lab section), Association(Student -- clinical lab section)
35	The system shall allow a Program Administrator or Nursing staff member to remove a student from a clinical lab section.	Association (Student -- clinical lab section)
36	The system shall allow a Program Administrator/Nursing Staff Member to move a student from one clinical lab section to another clinical lab section corresponding to the same clinical class.	Association (Student -- clinical lab section)
37	Program Administrators/Nursing Staff Members shall be able to cancel a clinical lab section only if there are no students registered for that clinical lab section.	Association (Student -- clinical lab section)



Table 5.16 Test data snippet for class relationship involved entities in functional text

Id	Functional Text	title
1	The student needs to use the system to registrar for courses.	Course Student
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Professor Course
7	The display shall have two regions: left 2/3 of the display is graphical right 1/3 of the display is a data table	region Graphical
7	The display shall have two regions: left 2/3 of the display is graphical right 1/3 of the display is a data table	region data table
9	The table side of the display shall be split into 2 regions: sequential and temporal.	region sequential
9	The table side of the display shall be split into 2 regions: sequential and temporal.	region temporal region
10	The top 1/4 of the table will hold events that are to occur sequentially.	Event sequentially occur event
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.	Event current time event
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Class clinical class
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	department class
21	A class shall be either a non-clinical class or a clinical class.	Class clinical class
21	A class shall be either a non-clinical class or a clinical class.	Class non clinical class
26	Program Administrators and Nursing Staff Members shall have the ability to specify which classes are required for a Program of Study	Program, Class
32	Program Administrators and Nursing Staff Members shall be able to add a new clinical lab section for an existing clinical	Clinical class, Clinical lab section
34	Program Administrators and Nursing Staff Members shall be able to add a student who has registered for a clinical class	Student Clinical class
34	Program Administrators and Nursing Staff Members shall be able to add a student who has registered for a clinical class	Clinical class, clinical lab section
34	Program Administrators and Nursing Staff Members shall be able to add a student who has registered for a clinical class	Student Clinical class section
35	The system shall allow a Program Administrator or Nursing staff member to remove a student from a clinical lab	Student clinical lab section
36	The system shall allow a Program Administrator/Nursing Staff Member to move a student from one clinical lab section to another clinical lab section corresponding to the	Student clinical lab section
37	Program Administrators/Nursing Staff Members shall be able to cancel a clinical lab section only if there are no	Student clinical lab section

Table 5.17 Model Predicted results snippet for identified class relationship involved entities

Id	Functional Text	title
1	The student needs to use the system to registrar for courses.	course student
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	professor course
7	The display shall have two regions: left 2/3 of the display is graphical right 1/3 of the display is a data table	region graphical
9	The table side of the display shall be split into 2 regions: sequential and temporal.	region temporal region
10	The top 1/4 of the table will hold events that are to occur sequentially.	event sequentially occur event
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.	event current time event
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	class clinical class
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	department class
21	A class shall be either a non-clinical class or a clinical class.	class clinical class
26	Program Administrators and Nursing Staff Members shall have the ability to specify which classes are required for a Program of Study	program, class
32	Program Administrators and Nursing Staff Members shall be able to add a new clinical lab section for an existing clinical class into the System.	clinical class, clinical lab section
34	Program Administrators and Nursing Staff Members shall be able to add a student who has registered for a clinical class to a clinical lab section for that class.	student clinical class section
36	The system shall allow a Program Administrator/Nursing Staff Member to move a student from one clinical lab section to another clinical lab section corresponding to the same clinical class.	student clinical lab section
37	Program Administrators/Nursing Staff Members shall be able to cancel a clinical lab section only if there are no students registered for that clinical lab section.	student clinical lab section

Second model will be identifying the class relationship type of that particular relationship that can identify in respective functional requirement text. Table 5.18 shows the Test data snippet used to test the second ML model and Table 5.19 shows the predicted result of it.

Table 5.18 Test data snippet for class relationship type in functional text

Id	Functional Text	Class Relationship Type
1	The student needs to use the system to registrar for courses.	Association
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Association
7	The display shall have two regions: left 2/3 of the display is graphical right 1/3 of the display is a data table	Generalization
7	The display shall have two regions: left 2/3 of the display is graphical right 1/3 of the display is a data table	Generalization
9	The table side of the display shall be split into 2 regions: sequential and temporal.	Generalization
9	The table side of the display shall be split into 2 regions: sequential and temporal.	Generalization
10	The top 1/4 of the table will hold events that are to occur sequentially.	Generalization
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.	Generalization
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Generalization
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	Association
21	A class shall be either a non-clinical class or a clinical class.	Generalization
21	A class shall be either a non-clinical class or a clinical class.	Generalization
26	Program Administrators and Nursing Staff Members shall have the ability to specify which classes are required for a Program of Study	Association
32	Program Administrators and Nursing Staff Members shall be able to add a new clinical lab section for an existing clinical	Association
34	Program Administrators and Nursing Staff Members shall be able to add a student who has registered for a clinical class	Association
34	Program Administrators and Nursing Staff Members shall be able to add a student who has registered for a clinical class	Association
34	Program Administrators and Nursing Staff Members shall be able to add a student who has registered for a clinical class	Association
35	The system shall allow a Program Administrator or Nursing staff member to remove a student from a clinical lab	Association
36	The system shall allow a Program Administrator/Nursing Staff Member to move a student from one clinical lab section to another clinical lab section corresponding to the	Association
37	Program Administrators/Nursing Staff Members shall be able to cancel a clinical lab section only if there are no	Association

Table 5.19 Model Predicted results snippet for identified class relationship type

Id	Functional Text	Class Relationship Type
1	The student needs to use the system to registrar for courses.	Association
3	The Professor needs to use the system to select the courses to teach for a semester, and must be able to request a course roster from the system.	Association
7	The display shall have two regions: left 2/3 of the display is graphical right 1/3 of the display is a data table	Association
9	The table side of the display shall be split into 2 regions: sequential and temporal.	Association
10	The top 1/4 of the table will hold events that are to occur sequentially.	Association
11	The bottom 3/4 of the table will hold events that occur according to its relevance to current time.	Association
18	Program Administrators and Nursing Staff Members shall be able to add clinical classes or sections to a sequence of classes	Association
20	Program Administrators and Nursing Staff Members shall be able to add new classes for the nursing department into the system.	Association
21	A class shall be either a non-clinical class or a clinical class.	Association
26	Program Administrators and Nursing Staff Members shall have the ability to specify which classes are required for a Program of Study	Association
32	Program Administrators and Nursing Staff Members shall be able to add a new clinical lab section for an existing clinical class into the System.	Association
34	Program Administrators and Nursing Staff Members shall be able to add a student who has registered for a clinical class to a clinical lab section for that class.	Association
36	The system shall allow a Program Administrator/Nursing Staff Member to move a student from one clinical lab section to another clinical lab section corresponding to the same clinical class.	Association
37	Program Administrators/Nursing Staff Members shall be able to cancel a clinical lab section only if there are no students registered for that clinical lab section.	Association

### 5.4.3 Generating usecase and class Diagram

In here after identifying relevant usecase and class diagram elements from the functional requirement text using NLP and ML models, particular diagram can be drowned with Plant UML tool. Generated diagram can evaluate using and comparing the actual diagram that used to test the program.

#### Generating Usecase diagram

Generated language code after deriving the use case elements as below in Figure 5.3. In their identified usecase diagram elements have been put together in a single language code to generate usecase diagram.

```

Edit Usecase Diagram Text
@startuml
left to right direction
skinparam packageStyle rectangle
title Use Case Diagram
actor registrar student
actor billing
actor professor course
actor registrar course
actor user course

rectangle System {

usecase (student course)
usecase (register course courses information billing)
usecase (course courses roster validation)

the billing system register for courses : Association
request course roster : Association
professor select courses to teach : Association
student register for courses : Association
registrar maintain student information : Association
}
@enduml

```

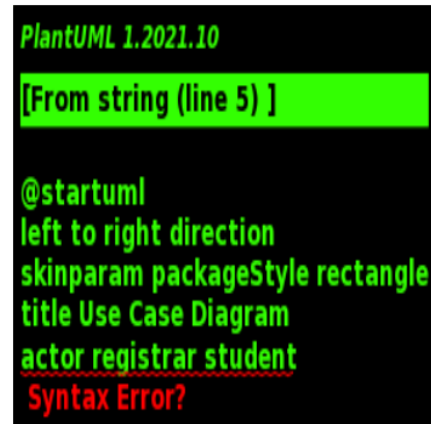


Figure 5.3 Program generated language code for usecase diagram

Syntax error would occur in generated code as Figure 5.3 and user can make changes to the code with referring identified actor, usecases and their relationship. In below Figure 5.4 shows the edited version of above Figure 5.3 language code.

```

Edit Usecase Diagram Text
@startuml
left to right direction
skinparam packageStyle rectangle
title Use Case Diagram
actor registrar
actor student
actor theBillingSystem
actor professor|
actor user

rectangle System {

usecase (student course)
usecase (register course courses information billing)
usecase (course courses roster validation)

theBillingSystem -- (register for courses) : Association
professor -- (request course roster) : Association
professor -- (select courses to teach) : Association
student -- (register for courses) : Association
registrar -- (maintain student information) : Association
}
@enduml

```

Figure 5.4 Edited language code for usecase diagram

Figure 5.5 shows the actual diagram that was given in case study and Figure 5.6 shows the generated usecase diagram according to above Figure 5.4 edited language code.

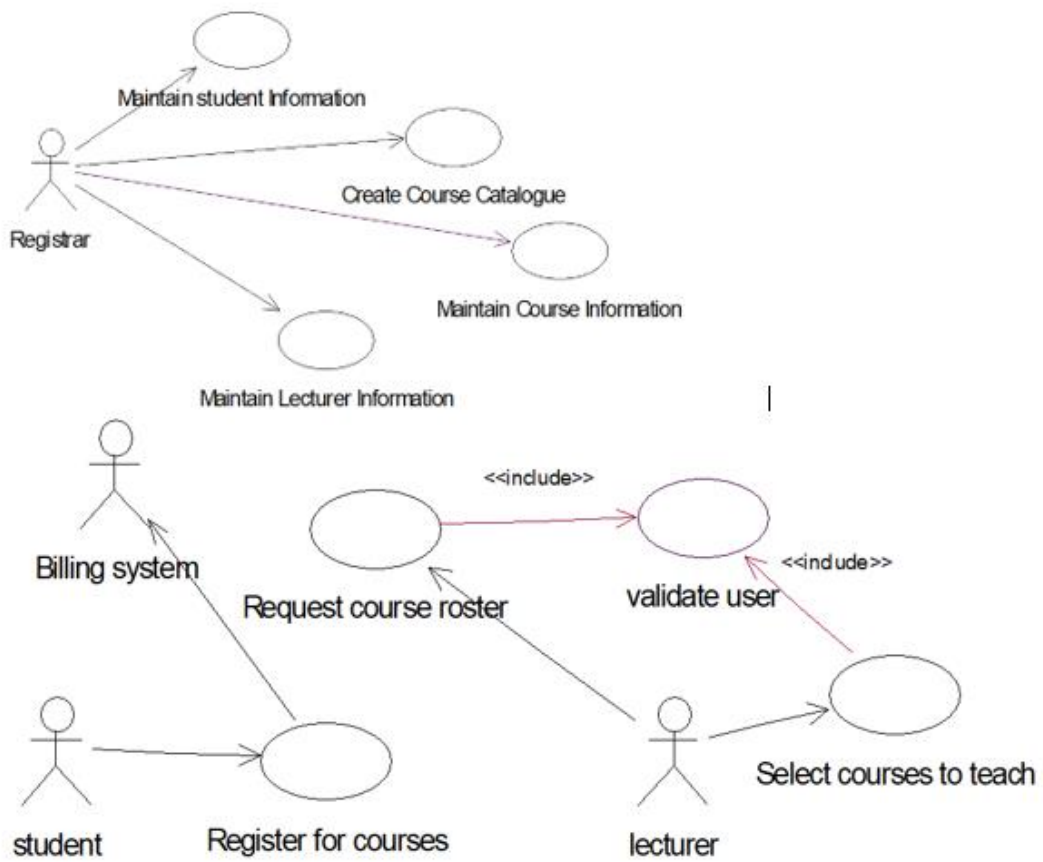


Figure 5.5 Actual use case diagram according to the functional text

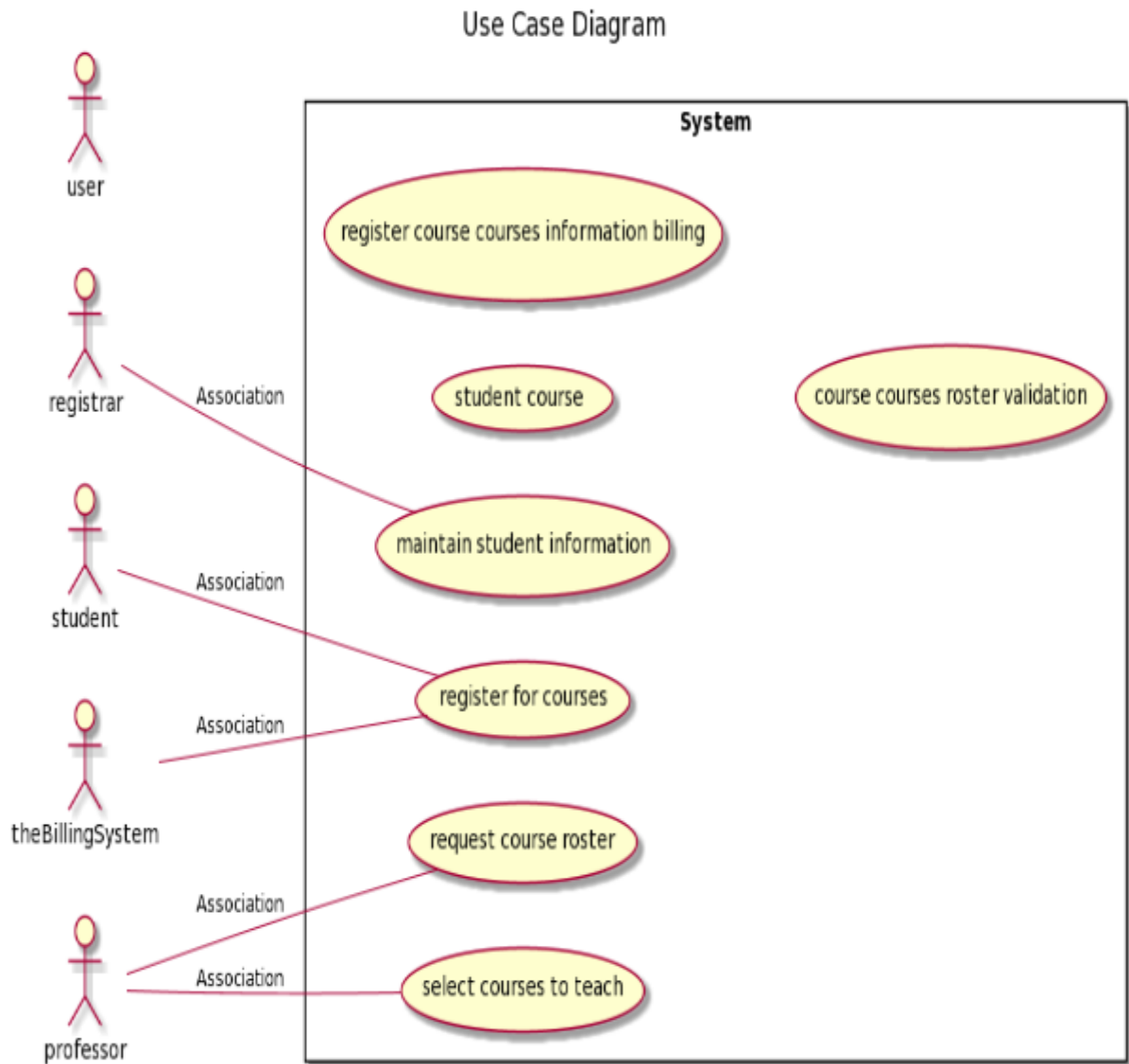


Figure 5.6 Generated usecase diagram using Plant UML tool

### Generating Class diagram

Generated language code after deriving the class diagram's elements as below in Figure 5.7. In their identified class diagram elements have been put together in a single language code to generate class diagram.

### Edit Class Diagram Text

```
@startuml
left to right direction
title Class Diagram
class student
class course

event equipment : Association
menu product : Association
professor course : Association
course student : Association
professor enterprise : Association
@enduml
```

PlantUML 1.2021.10

**[From string (line 7) ]**

```
@startuml
left to right direction
title Class Diagram
class student
class course
```

```
event equipment : Association
Syntax Error?
```

Figure 5.7 Program generated language code for class diagram

### Edit Class Diagram Text

```
@startuml
left to right direction
title Class Diagram
class student
class course

professor -- course : Association
course -- student : Association
@enduml
```

Figure 5.8 Edited language code for class diagram



After identifying syntax error and incorrect classes and their relationships user can edit the language code accordingly. Edited language code for above Figure 5.7 would be as above Figure 5.8

Figure 5.9 shows the actual diagram according to the given functional text and Figure 5.10 shows the generated class diagram according to above Figure 5.8 edited language code.

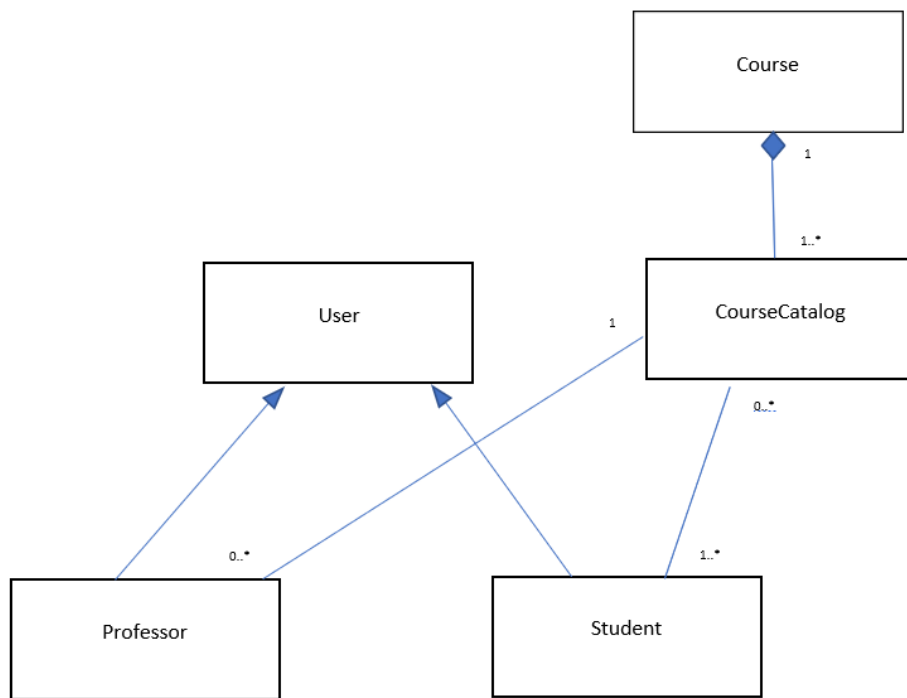


Figure 5.9 Actual usecase diagram according to the functional text

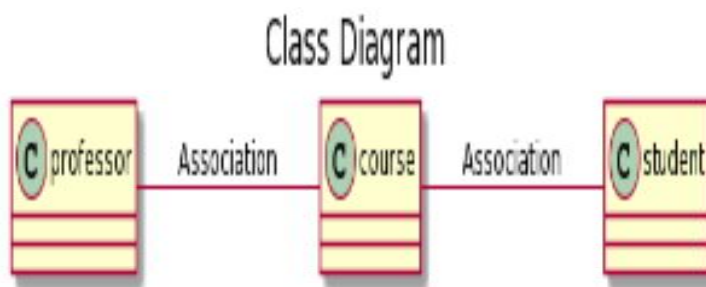


Figure 5.10 Generated class diagram using Plant UML tool

As a conclusion this project prototype has identified some important elements that have identified by the evaluator, in here it mentioning as actual elements or diagram for particular functional requirement text. If this project is domain specific most of element would have been identified the application itself. In general, this proposed diagram generating tool has identified and generated the use case diagram and class diagram up to a considerable extent which gives a good idea about the scenario.

This chapter started with describing collecting of dataset and dataset preparation where it described the data acquiring method from various sources for the both NLP and ML model training and validation. Then evaluation approach and selected different types of evaluators as classification metrics to evaluate the different models of the project along with the justifications for those selections. Evaluation and results were discussed deeply according to the identifying use case diagram element from requirement text and identifying class diagram element from requirement text. In there expected results against predicted results shown according to the snippets of the test dataset and also generated diagram using plant UML tool also be shown against the actual diagram which is expected according to the test functional text.

# CHAPTER 6

## CONCLUSION AND FUTURE WORK

The previous chapter presented the results of the evaluation process carried out on the System. This chapter will be focused on concluding the project by highlighting the achievement of the goal and objectives, the problems and challenges faced during the life cycle of the project. And also, in here the limitations of the project, identified future enhancements and closing remarks describe at the end of this chapter.

### 6.1 Conclusion

Comparing the research and studies conducted in this context, the generation of UML use case diagram and class diagram from the natural language text using natural language processing and machine learning technologies can still be considered a relatively new field. There have been several research attempts to generate automated use case diagrams and class diagrams using natural language processing and machine learning methods to extract the element of that diagram from a functional requirement text or user story text. Also, generating the use case and the class diagram remain the most challenging areas for finding the ability to analyze and understand business requirement text as a unit of interpretation to extract important elements.

There was good amount of work related to NLP techniques related to identify UML diagram elements from functional requirement text or user stories. The work represented by C. R. Narawita, K. Vidanage (2017) is quite similar to the present one but their approach was more towards with rule-based approach. According to specific sentence pattern they identifying usecase and class diagram elements from functional requirement text. They were used classification model to identify relationships type but that's not specifying what element would involve to that relationship. And also, they suggested to use regression and they believe that would be a better approach compared to classification.

There were some works that automatically generate conceptual models from a series of user stories in the form of OWL ontology, and our approach to automatically generating UML use case diagram from functional requirements text or user stories. The advantage of this

technology is the ability to reduce the ambiguity of software requirements specifications and to facilitate the work of the development team and product owner to generate automated design templates. The advantage of using automate UML use case and class diagram generator is that, it is easy to understand and helps the developer to interpret the functional requirements or user stories in one way so that the teams are really integrated in the design process. Also, this allows designers to save time as they can generate UML use case and Class models from a series of functional requirement or user stories in a short time.

In this research, the developed prototype can read and fully analyze the functional requirements provided in English language texts. It can also automatically generate use case diagram and class diagrams. In our system early stage uses NLP technologies such as tokenization and POS tagging to interpret system specifications based on a predetermined set of syntactic heuristic rules. Then, our proposed system includes the ability to analyze and understand input scenarios with the help of a machine learning model with a classification and also Recurrent neural network-based Sequence to Sequence Model.

We have taken the advantage of naïve Bayes classifier in order to identify multiple attributes of diagram. The features for each attribute set are different from other. Hence, we would calculate independent features for all. Based on this assumption we compute separate features and use naïve Bayes for extraction of key terms. And also, Recurrent neural network-based Sequence to Sequence Model used to identify relationships among actors to use cases, use case to use case and class relationships. Using the training dataset we prepared, a classification model can be designed and pre-process the data for extracting the key terms such as actors, use cases, class and their relationship.

In the initial process, the proposed system faced some challenges in research and development and had to be developed throughout the experiments with different approaches. The author has done a lot of experimentation and brought it to its present state. Currently the proposed system has its own ingenuity to extract the essence of UML use case and class diagrams and can generate use case and class diagrams according to the provided requirement text with customization. The main objective of this project was to automate generation of use case diagram and class diagram from the user input text scenario to reduce the time, and cost factors

for both system users and business analyst. That input text would be functional requirement text or user stories in the agile point of view.

Identifying class attributes and multiplicity for classes is very important features in generating the class diagram. Furthermore, it is important to correctly identify include and extend relationships in the use case diagram. However, due to the time limit of the research and the complexity of the task, the author has placed it as a future improvement with the machine learning model, which requires a high level of intelligence and high accuracy and a good amount of data for training. The better improved version of a machine learning or deep learning model should be considered in this regard.

Moreover, the proposed method provides a static and dynamic view for a given business requirement by generating a class diagram and use case diagram where user can edit the diagram with Plant UML language code. In relation to the evaluation and testing performed, the system demonstrates the ability to generate use case and class diagram according to the input text of the functional requirement or set of user stories. With the use of NLP and ML technologies researched and implemented by the author, the relevant system will give adequate results in a reasonable time. But the results can prove that the approach is successful and adaptable to a variety of business situations. This is simple to prove that the system is pretty much a good product. The system has the ability to identify business decisions in words, by analyzing the user's writing style.

In conclusion, the author has done an in-depth analysis and improvements in the field of generating the use case and class diagram against business requirement. By fine-tuning and preparing more data for different set of business domains this research may eventually lead to a commercial product, which will certainly help the system users to get a quick and rough overview regarding to the system to be developed.

## **6.2 Future Work**

This system can generate use case diagram and class diagram with identifying relationship and it's involved component. It does not have the capability to identify relationship types like include, extend and generalize relationships in use case diagram, and Aggregation

Composition, Generalization in class diagram. Therefore, we hope to provide the ability to reuse existing use cases to reduce the efforts required to define the use cases and class relationship of the system. Our goal, on the other hand, is to extend our system to automatically identify other features such as class diagram multiplicity and class attribute. With this phase it couldn't achieve as reason of Limited amount of data and Domain variation among data instances. Improving this sequence-to-sequence learning approach with significant amounts of data may be targeted for specific domain data in the future.

## REFERENCES

- ✓ Adhav, V., Ahire, D., Jadhav, A. and Lokhande, D. (2015). ‘Class Diagram Extraction from Textual Requirements Using NLP Techniques’. *IOSR Journal of Computer Engineering*, 17(2), pp.27–29.
- ✓ Ayana, Shen, S.-Q., Lin, Y.-K., Tu, C.-C., Zhao, Y., Liu, Z.-Y. and Sun, M.-S. (2017). Recent Advances on Neural Headline Generation. *Journal of Computer Science and Technology*, 32(4), pp.768–784.
- ✓ Azzazi, A. (2017). ‘A Framework using NLP to automatically convert User-Stories into Use Cases in Software Projects’. *IJCSNS International Journal of Computer Science and Network Security*, 17(5), p.71. Available at: [http://paper.ijcsns.org/07\\_book/201705/20170510.pdf](http://paper.ijcsns.org/07_book/201705/20170510.pdf) (Accessed: 10 Sep. 2020).
- ✓ Bahdanau, D., Cho, K. and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv.org*. Available at: <https://arxiv.org/abs/1409.0473>.
- ✓ Bajwa, I. S. & Choudhary, M. A. (2006) “Natural language processing based automated system for uml diagrams generation,” in *The 18th Saudi National Computer Conf. on computer science (NCC18)*. Riyadh, Saudi Arabia: The Saudi Computer Society (SCS), Riyadh, Saudi Arabia, 2006, pp. 1-6.
- ✓ Barba, P., Lexalytics, (2020). ‘Machine Learning (ML) for Natural Language Processing (NLP)’. Available at: <https://www.lexalytics.com/lexablog/machine-learning-natural-language-processing> (Accessed: 29 Aug 2020).
- ✓ Bhagat, S., Kapadni, P., Kapadnis, N., Patil, D. and Baheti, M. (2012), ‘Class Diagram Extraction Using NLP’. *International Journal of electronics, Communication & Soft Computing Science & Engineering*, p.1 Available at: <http://www.ijescscse.org/papers/SpecialIssue/comp2/190.pdf> (Accessed: 10 Sep. 2020).
- ✓ Bradbury, J., Merity, S., Xiong, C. and Socher, R. (2016). Quasi-Recurrent Neural Networks. *arXiv:1611.01576 [cs]*. Available at: <https://arxiv.org/abs/1611.01576> [Accessed 28 Aug. 2021].
- ✓ Byeon, W., Breuel, T., Raue, F. and Liwicki, M. (2015). Scene Labeling with LSTM Recurrent Neural Networks. , pp.3547–3555.
- ✓ Cho, K., Merriënboer, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv.org*. Available at: <https://arxiv.org/abs/1406.1078>.
- ✓ Chopra, S., Auli, M. and Rush, A.M. (2016). Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp.93–98.
- ✓ Computer Notes. (2013). ‘Write a Note on Software Design Phases’. Available at: <https://ecomputernotes.com/software-engineering/write-a-note-on-software-design-phases> (Accessed: 20 Mar. 2021).
- ✓ Deeptimahanti, D. and Babar, M., (2009). ‘An Automated Tool for Generating UML Models from Natural Language Requirements’. *2009 IEEE/ACM International Conference on Automated Software Engineering*, pp 680-682. doi:10.1109/ASE.2009.48.
- ✓ Deeptimahanti, D.K. and Sanyal, R. (2011). ‘Semi-automatic generation of UML models from natural language requirements’. *Proceedings of the 4th India Software Engineering Conference on - ISEC '11*, pp. 165-174. doi:10.1145/1953355.1953378.
- ✓ Donahue, J., Hendricks, L.A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K. and Darrell, T. (2014). Long-term Recurrent Convolutional Networks for Visual Recognition and Description. *arXiv.org*. Available at:

- <https://arxiv.org/abs/1411.4389> [Accessed 7 Aug. 2021].
- ✓ Elallaoui, M., Nafil, K. and Touahni, R., (2018). ‘Automatic Transformation of User Stories into UML Use Case Diagrams using NLP Techniques’. *Procedia Computer Science*, 130, pp.42-49.
  - ✓ Greff, K., Srivastava, R.K., Koutnik, J., Steunebrink, B.R. and Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10), pp.2222–2232. Available at: <https://arxiv.org/pdf/1503.04069.pdf>.
  - ✓ Guru99.com. (2019). ‘Functional Requirements vs Non Functional Requirements: Key Differences’. Available at: <https://www.guru99.com/functional-vs-non-functional-requirements.html> (Accessed: 1 Sep. 2020).
  - ✓ Hamza, Z. and Hammad, M., (2019). ‘Generating UML Use Case Models from Software Requirements Using Natural Language Processing’. 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO).
  - ✓ He, K., Zhang, X., Ren, S. and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv.org*. Available at: <https://arxiv.org/abs/1512.03385>.
  - ✓ Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), pp.1735–1780.
  - ✓ Holland, K. (2018). ‘What is a Class Diagram?’ *Medium*. Available at: <https://medium.com/@katie.holland.runs.fast/what-is-a-class-diagram-c74a2129e66c> (Accessed: 20 Mar. 2021).
  - ✓ Ibrahim, M. and Ahmad, R., (2010). ‘Class Diagram Extraction from Textual Requirements Using Natural Language Processing (NLP) Techniques’. 2010 Second International Conference on Computer Research and Development, pp. 200-204. doi:10.1109/ICCRD.2010.71.
  - ✓ Jaiwai, M. and Sammapun, U. (2017). ‘Extracting UML Class Diagrams from Software Requirements in Thai using NLP’, pp. 1-5. doi: 10.1109/JCSSE.2017.8025938.
  - ✓ Jiang, Y. and Bansal, M. (2018). Closed-Book Training to Improve Summarization Encoder Memory. *arXiv:1809.04585 [cs]*. Available at: <https://arxiv.org/abs/1809.04585> [Accessed 28 Aug. 2021].
  - ✓ Jordan, M.I. (1997). Serial Order: A Parallel Distributed Processing Approach. *Neural-Network Models of Cognition - Biobehavioral Foundations*, pp.471–495.
  - ✓ Jozefowicz, R., Zaremba, W. and Sutskever, I. (2015). An Empirical Exploration of Recurrent Network Architectures. *proceedings.mlr.press*. Available at: <http://proceedings.mlr.press/v37/jozefowicz15.html>.
  - ✓ Karpathy, A., Johnson, J. and Fei-Fei, L. (2015). Visualizing and Understanding Recurrent Networks. *arXiv:1506.02078 [cs]*. Available at: <https://arxiv.org/abs/1506.02078> [Accessed 8 Aug. 2021].
  - ✓ Klein, G., Kim, Y., Deng, Y., Senellart, J. and Rush, A.M. (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation. *arXiv.org*. Available at: <https://arxiv.org/abs/1701.02810> [Accessed 25 Aug. 2021].
  - ✓ Kumar, S.K. (2014). ‘Generation of UML class diagram from software requirement specification using natural language processing’. Available at: <https://www.semanticscholar.org/paper/Generation-of-UML-class-diagram-from-software-using-Kar/cc3acd603c865de6af012f0b1ab803ad86c88402> [Accessed 08 Sep. 2020].
  - ✓ Lei, T., Zhang, Y., Wang, S.I., Dai, H. and Artzi, Y. (2018). Simple Recurrent Units for Highly Parallelizable Recurrence. *arXiv:1709.02755 [cs]*. Available at: <https://arxiv.org/abs/1709.02755> [Accessed 28 Aug. 2021].
  - ✓ Madanayake R. S., (2019) ‘Transformation of Requirement Techniques to Reduce Duplication of Work in Methodologies’, MPhil Thesis, University of Colombo School



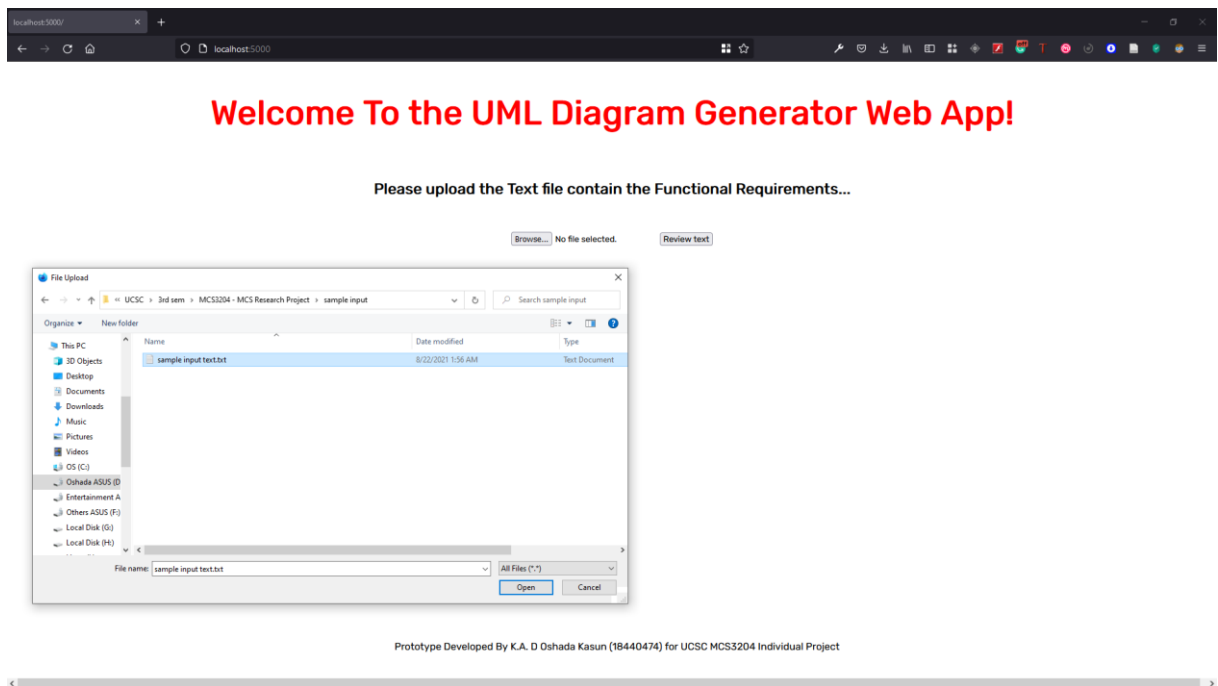
- of Computing. doi: 10.13140/RG.2.2.11058.22728.
- ✓ Moldovan, Dan & Surdeanu, Mihai. (2002). ‘On the Role of Information Retrieval and Information Extraction in Question Answering Systems’. pp. 129-147. doi:10.1007/978-3-540-45092-4\_6.
  - ✓ More, P. and Phalnikar, R. (2012). ‘Generating UML Diagrams from Natural Language Specifications’. *International Journal of Applied Information Systems*, 1(8), pp.19–23. doi: 10.5120/ijais12-450222.
  - ✓ Narawita, C & Vidanage, K. (2016). ‘UML generator - an automated system for model driven development’. pp. 250-256. doi:10.1109/ICTER.2016.7829928.
  - ✓ Nasiri, S., Rhazali, Y., Lahmer, M. and Chenfour, N. (2020). ‘Towards a Generation of Class Diagram from User Stories in Agile Methods’. *Procedia Computer Science*, 170, pp.831–837. doi: 10.4018/978-1-7998-3661-2.ch008.
  - ✓ Osman, M., Alabwaini, N., Jaber, T. and Alrawashdeh, T., (2019). ‘Generate use case from the requirements written in a natural language using machine learning’. 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT). Available at: <https://ieeexplore.ieee.org/document/8717428> (Accessed 5 Sep. 2020).
  - ✓ Pereira, A., 2018. ‘Using NLP to generate user stories from software specification in natural language’, Universidade Federal do Paraná. Setor de Ciências Exatas. Programa de Pós-Graduação em Informática, Available at: <https://acervodigital.ufpr.br/handle/1884/58882> (Accessed 6 Sep. 2020).
  - ✓ PlantUML.com. (n.d.). Class Diagram syntax and features. Available at: <https://plantuml.com/class-diagram> (Accessed: 10 March 2021).
  - ✓ PlantUML.com. (n.d.). Use case Diagram syntax and features. Available at: <https://plantuml.com/use-case-diagram> (Accessed: 10 March 2021).
  - ✓ Pradhan, S. and Longpre, S. (2016). Exploring the Depths of Recurrent Neural Networks with Stochastic Residual Learning.
  - ✓ Rachiele, G. (2018). ‘Tokenization and Parts of Speech(POS) Tagging in Python’s NLTK library’. [Medium]. Available at: <https://medium.com/@gianpaul.r/tokenization-and-parts-of-speech-pos-tagging-in-pythons-nltk-library-2d30f70af13b>. (Accessed: 07 May. 2021).
  - ✓ Rush, A.M., Chopra, S. and Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization. arXiv.org. Available at: <https://arxiv.org/abs/1509.00685> [Accessed 17 Aug. 2021].
  - ✓ Sutskever, I., Vinyals, O. and Le, Q.V. (2014). Sequence to Sequence Learning with Neural Networks. arXiv.org. Available at: <https://arxiv.org/abs/1409.3215>.
  - ✓ Vachharajani, V. and Pareek, J. (2014). ‘A Proposed Architecture for Automated Assessment of Use Case Diagrams’. *International Journal of Computer Applications*, 108(4), pp.975–8887. doi: 10.5120/18902-0193.
  - ✓ Vemuri, S., Chala, S. and Fathi, M., 2017. ‘Automated use case diagram generation from textual user requirement documents’. 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE). doi:10.1109/CCECE.2017.7946792.
  - ✓ Visual-paradigm.com. (2019). ‘What is Use Case Diagram?’ Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/> (Accessed: 20 Mar. 2021).
  - ✓ Whitney, E., CODE Magazine (2020). ‘Introduction to Gathering Requirements and Creating Use Cases’. Available at: <https://www.codemag.com/Article/0102061/Introduction-to-Gathering-Requirements-and-Creating-Use-Cases> (Accessed: 25 Aug 2020).
  - ✓ Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H. and Stevens, K. (2016). Google’s

- Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv.org. Available at: <https://arxiv.org/abs/1609.08144>.
- ✓ www.aha.io. (n.d.). 'User Stories vs. Requirements'. Available at: <https://www.aha.io/blog/user-stories-vs-product-requirements> (Accessed: 1 Sep. 2020).
  - ✓ Zakarya, M., Alqaralleh, B., Alemerien, K., Malek, Z., Alksasbeh and Alramadin, T. (2017). 'An Automated Use Case Diagrams Generator From Natural Language Requirements', 95(5), pp.1182-1190.

# Appendix A

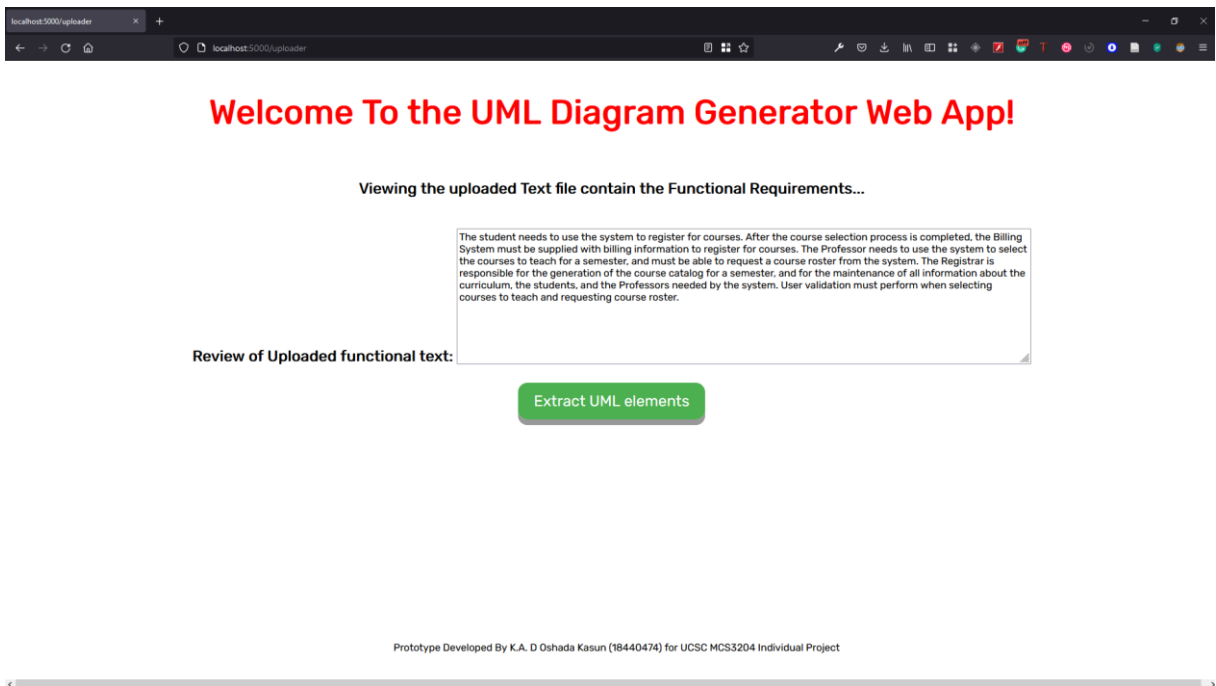
Following figure illustrate the prototype screenshots with brief description with steps to followed.

Initial page would be as below where user can upload text file containing the functional requirement.

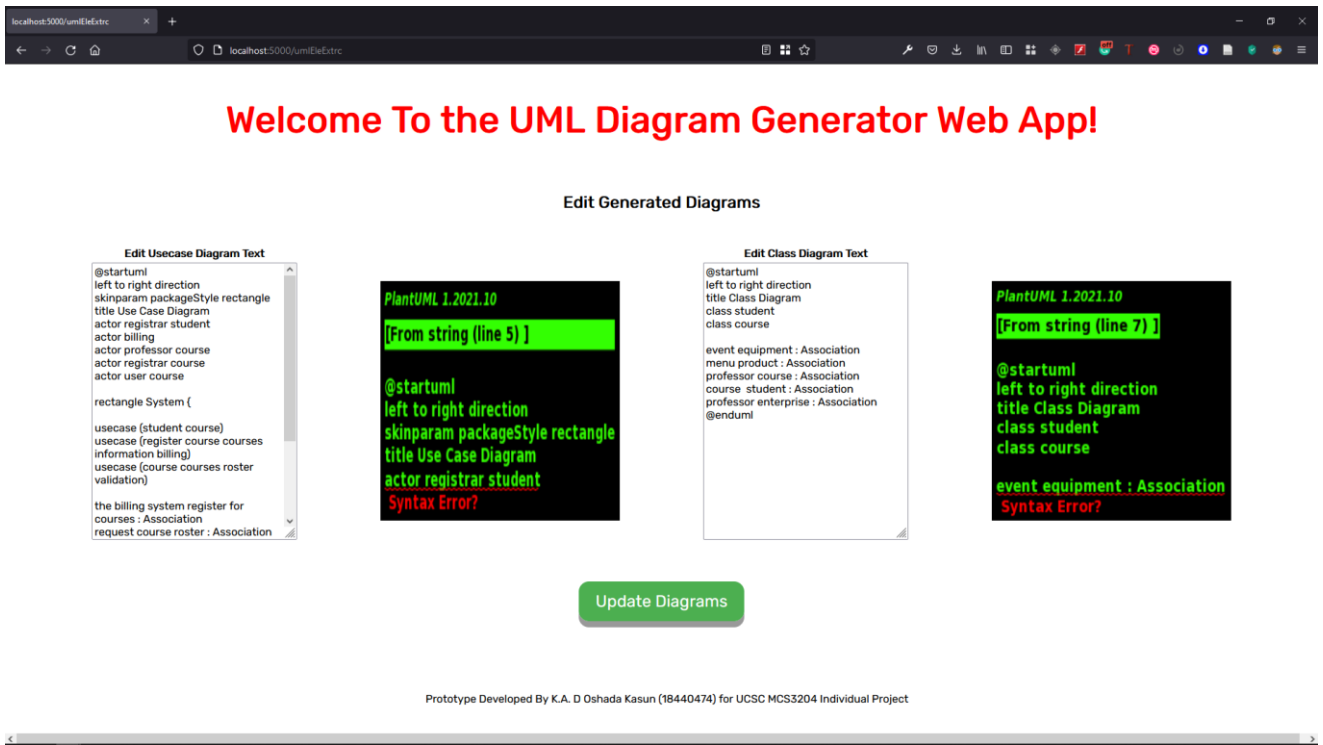




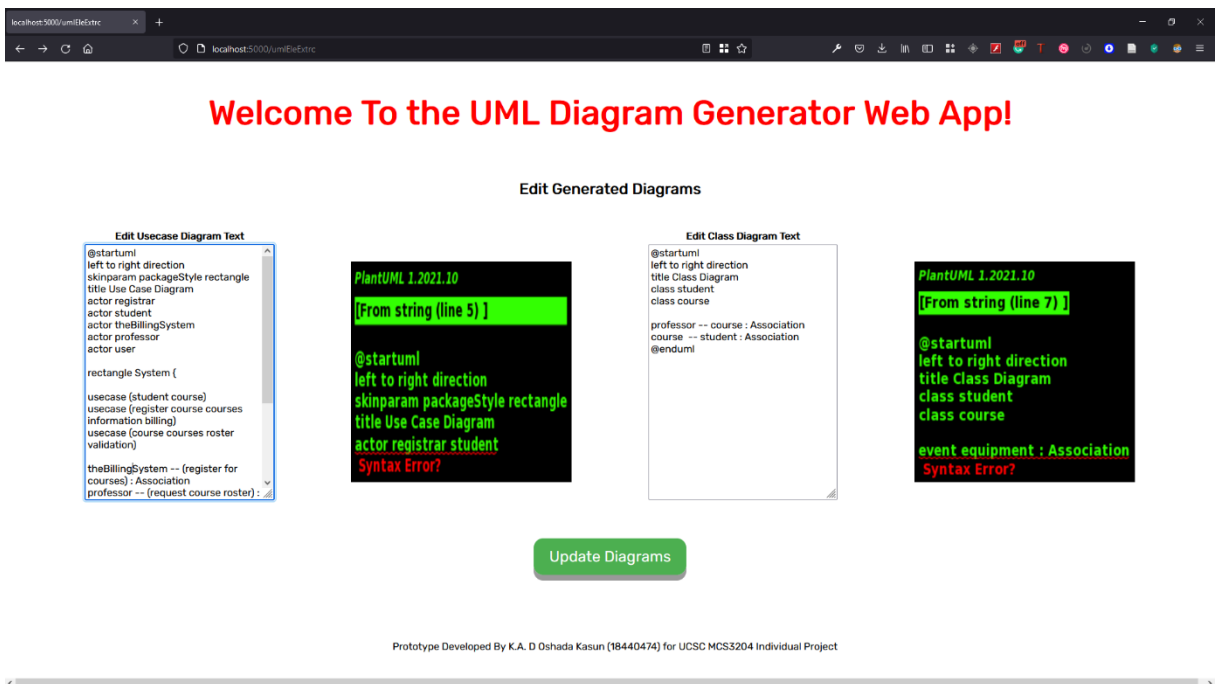
After uploading the text file user can review the text containing and can edit if needed.



After proceeding with the 'Extract UML elements' user can view the identified elements of each usecase and class diagram elements. And also user able to view the generated language code for each diagram and see whether generated language code is syntactically correct or where it has syntax errors if there any.



User can edit each diagram language code with removing or adding elements if there need. Following figure showing the edited code of above figure which is generated by the prototype.



After proceeding with 'Update Diagram' user will be able to view the usecase and class diagram generated accordingly.

localhost:5000 umlEdit

localhost:5000 umlEdit

## Welcome To the UML Diagram Generator Web App!

Generated Diagrams

Use Case Diagram

Class Diagram

Prototype Developed By K.A. D Oshada Kasun (18440474) for UCSC MCS204 Individual Project

If there any error with generating diagram that also will show with mentioning the syntax error with the line of the code as following figure.

localhost:5000 umlEdit

localhost:5000 umlEdit

## Welcome To the UML Diagram Generator Web App!

Generated Diagrams

```

PlantUML 1.2021.10
[From string (line 17) ]
@startuml
left to right direction
skinparam packageStyle rectangle
title Use Case Diagram
actor registrar
actor student
actor theBillingSystem
actor professor
actor user

rectangle System {
usecase (student course)
usecase (register course courses information billing)
usecase (course courses roster validation)
theBilling System .. (register for courses) : Association
Syntax Error?

```

Class Diagram

Prototype Developed By K.A. D Oshada Kasun (18440474) for UCSC MCS204 Individual Project

User can right click on each image and view each diagram in separate tab to view and download each diagram in original size. And also the particular URL for generated diagrams can be seen after opening it with separate tab.

