International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12– No.34, October 2020– www.ijais.org

Automated Use Case Diagram Generation with Non-functional Requirements using Neural Network

Israa Abdulruof Othman Ahmed
Elnasr Technical College
Information Technology Program
Khartoum, Sudan

Mubarak Elamin Elmubarak Daleel
Alzaeim Alazhari University
College Of Computer Science Information Technology- Khartoum, Sudan

ABSTRACT
The Unified Modeling Language (UML) is an excellent and well known powerful recognized leading diagrammatic modeling language. Recently, there have many efforts addressing automating these models, such as the Use Cases model where natural languages are used and giving the system a set of rules to be able to extract the actors and use cases from textual requirements. These rules confirm that all user requirements are included there in .In this paper a model has been developed using natural languages in addition to neural networks where the network is trained in the way of extracting Functional and nonfunctional requirements, which helps in producing more effective models that minimizing the time and effort of both the user and the analyst.

Keywords
Neural Network – UML - Machine learning - Use Cases Model - Nonfunctional Requirements.

1. INTRODUCTION
UML is used in scientific research problems and to design model [1, 2]. UML is broadly accepted as modeling language by academia and industry people to model a software system [3]. Use Case modelling provides a graphical representation of the software system's requirements. It is a useful tool for requirements elicitation. The key elements in a use case model are actors (external entities), and the use cases themselves. A use case is a unit of service or functionality (a requirement), in the system. A use case is not a program or process, or function in outline.

Use case models are relatively easy to discuss the correctness of a use case model with a non-technical person (such as a customer), because they are simple both in concept and appearance.

Use case modeling has effectively become a practicable analysis technique with the publication of Ivar Jacobson's (1991) book "Object-oriented software engineering: a use case driven approach". Jacobson has continued to enhance this approach to system analysis up to date, and it has now been formalized as part of the UML, although, use case modeling is not very different in its strategy and purpose from earlier techniques such as structured viewpoint analysis [4].

Machine learning (ML) is ideal for exploiting the hidden opportunities in big data. Machine learning is a type of artificial intelligence (AI) that allows software applications to become more accurate in predicting outcomes without being explicitly programmed.

The basics of machine learning is to build algorithms that can use statistical analysis and take input data to predict an output value within an acceptable range. Machine Learning extracts value from

Disparate and big data sources with far less reliance on human direction.

It runs at machine scale and data driven and it is well suited to the complexity of dealing with disparate data sources and the huge amounts of data and variety of variables involved. Furthermore, unlike traditional analysis, machine learning thrives on growing datasets. The more data are fed into a machine learning system, the more it can apply and learn the results to insights. [5]

2. OBJECTIVE
To develop a framework that helps ensure the comprehensiveness of the user requirements generated by the automatic use cases model, which leads to the production of more efficient software systems.

3. RELATED WORKS
There are many existing NLP based automated tools to support requirement analysis phase of software development. Following is a brief survey of such tools:

In the researchers conducted at the Del institute of technology (DIT) and Heriot-Watt University (HWU) [6], complex sentences were split into independent sentences of standard SVO structure using NL rules (for example conjunctions are used as identifiers). In case of the tool created by DIT, first word of each sentence other than a pronoun is considered as an actor and the rest as use case. It works with only simple and active sentences. In the latter case, nouns are considered as classes and noun-noun forms are considered as class property. Passive sentences are converted to active using the verb tense.

Also LESSA [7] produces small diagrams for each sentence and uses similar approach. These diagrams are combined to form the final use case diagram.

CM-BUILDER [7], NL-OOPS [8] and R-Tool [9] are typically NLP based Computer Aided Software Engineering (CASE) tools. They try to produce a rough cut class diagrams from the user requirement document (URD), which has to be further corrected by the analyst to produce actual class diagrams. R Tool generates a frequency distribution and extracts all the nouns in the document and then the most frequent nouns become actors. Verbs associated with them become use cases and the rest become attributes and so on. In

our paper the attributes are connected with noun to form non-functional requirements.

NL-OOPS and CM-BUILDER

These try to create hierarchical trees of words in each sentence after extraction of word features. They apply grammatical constraints to figure out the relationships. Part of speech (POS) tags help identify the words as events, objects, entities, etc., then these words are ontologically analyzed to figure out the instances and relationships.

The most common method used to extract word features is by applying certain NL rules. It is evident that splitting compound sentences avoids loss of information as well as ambiguity.

4. PROPOSD METHOD

The figure below outlines the steps of the model

After the classifiers are trained, they create a multinomial NB distribution of the training data. Classification process takes place in two parts. In the first part, group (a) of the preprocessed sentences are sent to the actor classifier one by one to classify these set of nouns into one of the predefined subject classes. In the second part, from table 1 ignoring probable subjects, probable objects from group (b) are sent to use case classifier to predict an object for each row.

Classification results are collected in Verb + Object combinations for each row and Verbs are retained as they are. These combinations form the use cases.

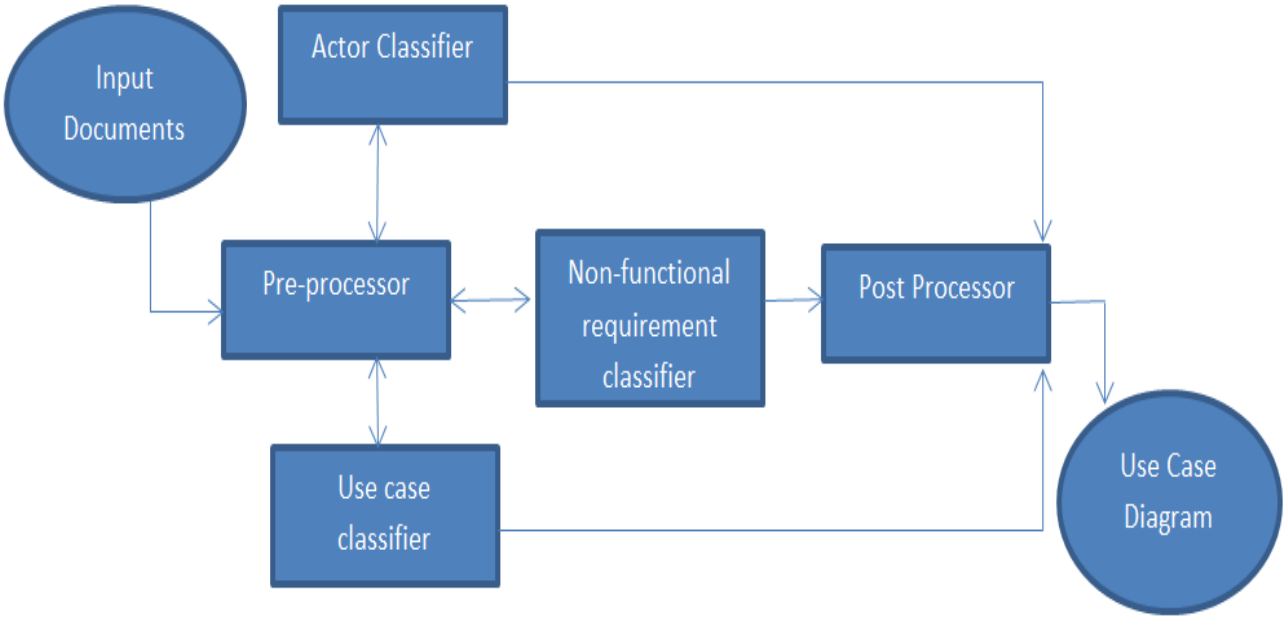


Fig 1: Process Flow of the Proposed Approach

These process steps can be summarized as follows.

4.1 Input User Requirement

In this phase, the sentence is entered by the user. It is entered in the English language, taking into account the grammar and correctness of the entry.

4.2 Pre- processing

Following are the individual tasks performed by the preprocessor.

4.2.1 Tokenizing

User requirement is split into a multidimensional list such that each word in the sentence, forms a column and each sentence forms a row.

4.2.2 POS tagging

Sentences are syntactically analyzed by the algorithm to identify the POS of each word. Then it is bundled with the word into a tuple.

4.2.3 Extraction of nouns and verbs

Algorithm filters out only the verbs and nouns in each sentence maintaining the same order. However unlike group (b), for group (a) of preprocessing only nouns are extracted.

4.2.4 Splitting

Splitting sentence into parts with individual use cases.

4.3 Classifier

Super vector Machine classifiers models a probabilistic classifier based on the features of word present in several classes. Then a text is classified based on its posterior probability belonging to several classes depending on the presence of words [10].

4.4 Nonfunctional requirement classifier

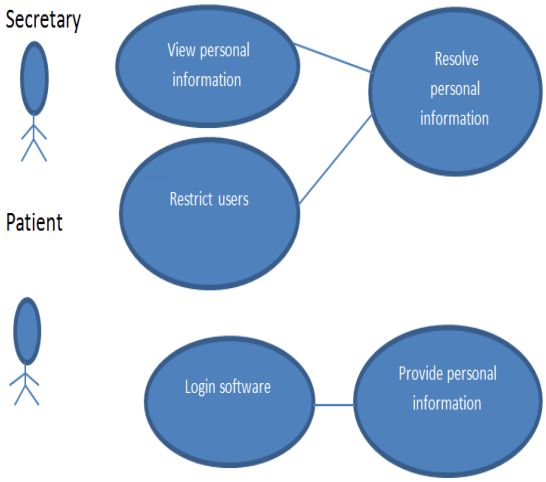
In this stage the same algorithm is used in stage 3 with a slight difference that each name will be associated with the adjective in the same row and this will form the non-functional requirement

4.5 Post processing

- Adding actors to the diagram
- Adding use cases to the diagram
- Adding nonfunctional requirement to the diagram
- Mapping the relationship

Table [1].Case Study

<table>
<tr><th>Phase</th><th>Task</th><th colspan="2"></th></tr>
<tr><td>Input</td><td>Read</td><td colspan="2">Patient will have to log in to the software to
Provide personal information. Secretary can view

Personal information and also resolve complaints.
Secretary can also restrict number of users.
Software should be available every day.</td></tr>
<tr><td rowspan="10">Pre -processing</td><td></td><td>Group (a)</td><td>Group (b)</td></tr>
<tr><td>Tokenizing</td><td>…..</td><td>…..</td></tr>
<tr><td>Pos</td><td>…..</td><td>…..</td></tr>
<tr><td>Extract feathers</td><td>…..</td><td>…..</td></tr>
<tr><td>Split sentences</td><td>…..</td><td>…..</td></tr>
<tr><td>Remove</td><td>…..</td><td>…..</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td>[' Patient '] ['software'] [' personal information '] [secretary '] [' personal information '] ['complaints'] [' secretary '] ['number'] ['users']</td><td>***
[‘Patient’] log in
[‘software’]
[--]provide [‘personal information’]
[‘secretary ‘]view
[‘personal information ‘][--]
restrict
[‘Number of users.’]
[--] resolve [‘complaints’]</td></tr>
<tr><td rowspan="2">classifier</td><td></td><td>Actor classifier</td><td>User classifier</td></tr>
<tr><td></td><td>[‘Patient’]
[‘secretary ‘]
[‘secretary ‘]</td><td>login['software']
provide[' persona l information ']
***
view[' personal information ']
resolve['complaints']
***
restrict['number', 'users']</td></tr>
<tr><td rowspan="2">Nonfunctional requirement classifier</td><td rowspan="2">Software [should’] available</td><td>Noun classifier</td><td>adjective classifier</td></tr>
<tr><td>software</td><td>available</td></tr>
<tr><td rowspan="4">Post processing</td><td>Add actor</td><td colspan="2"></td></tr>
<tr><td>Add use case</td><td colspan="2" rowspan="3">Software should be available every day
 
</td></tr>
<tr><td>Map relationship</td></tr>
<tr><td>Add Nonfunctional requirement</td></tr>
</table>

5. RESULTS AND EVALUATION

A confusion matrix states the accuracy of the solution to a classification problem. Given m classes, a confusion matrix is an m x m matrix where entry ci,j represents the tuples from D that were assigned to class Cj but where the correct class is Ci. Definitely, the best solutions will have only zero values outside the diagonal [11]. The confusion matrix is a useful tool for analyzing how well the classifier can recognize tuples of different classes. Given m classes, a confusion matrix is a table of at least size m by m [12].

Table 1 shows a confusion matrix for height classification. In confusion matrix, the columns represent the predicted classifications, and the rows represent the actual (true) classifications [13].

In a multiclass classification, a confusion matrix is necessary to be observed and changed into table of confusion as shown in Table 2.

Table [2]. Table of Confusion

TRUE POSITIVE (TP)	FALSE NEGATIVE (FN)
FALSE POSITIVE (FP)	TRUE NEGATIVE (TN)

Precision= TP/TP+FP
Recall (TP Rate) = TP/TP+FN
F-Measure= 2*(Precision* Recall/ Precision+ Recall)
Success Rate = TP+TN/ P+N
Where: P = TP + FN and N = FP + TN

In this research, a system of ―Confusion Matrix for Accuracy‖ is made by researchers based on the formulas of (1), (2), (3), and (4). The system is built using Winpython-3.5.2. The purpose is to observe the confusion matrix, investigate the table of confusion, and measure the accuracy of the framework in precision, recall, F-measure, and success rate.

For the final evaluation, three different user requirements documents from the same domain were used. None of these documents were used to train the classifiers nor to test during the development phase. Out of these documents, the first had 5 sentences; other had 10 and 15 sentences respectively with an average sentence length of 10 words.

6. RECOMMENDATIONS

1. The functionality of the conducted research is domain specification, so we expect it to be enhanced by future researchers.
2. The developed framework is about use case diagram and the work needed to be expanded in order to include the other UML models.

7. CONCLUSION AND FUTURE WORKS

In conclusion, this paper proposed a model using natural languages in addition to neural networks to generate the use case diagram automatically, but there are still some efforts that can be made in order to generate the rest of UML models like: sequence diagram, component diagram …etc.

In this model the network is trained in the way of extracting Functional and nonfunctional requirements, a confusion matrix have been used for evaluation, the proposed algorithm performs with recall of 90%, an accuracy of 87%, and precision of 80%.

8. REFERENCES

[1] Kovacevic. S. UML and User Interface Design, inUML'98. France -Mulhouse, 1998.

[2] Ansari, G.A. "A Modeling and Detection of Dead Lock in Early Stage of System Using UML", International Journal of Computer Applications (IJCA). pp. 16-20, Vol. 39 No. 9/ February 2012.

[3] M. Flower, UML distilled (3d Edition), Addison wesly, 2003.

[4] MSc-IT Study Material January Edition 2011 Computer Science Department. University of Cape Town

[5] Guido, S, Muller, A. (October 2016). Introduction to Machine Learning with Python: A Guide for Data Scientists.USA: O'Reilly Media.

[6] A Hutagaol, D Simarmata, J Manihuruk E M Sibarani, "Actor and Use Case Extraction from Text-Based Requirement Specification".

[7] S Aithal, P Desai S Vinay, "An Approach towards Automation of Requirements Analysis," in Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I IMECS 2009, Hong Kong, 2009.

[8] R Garigliano L Mich, "NL-OOPS: A Requirements Analysis tool based on Natural Language Processing," in Conference on Data Mining, 2002, Vol. 3.

[9] R Gaizauskas H M Harmain, "CM –Builder: An Automated NLP-based CASE Tool," in the Fifteenth IEEE International Conference on Automated Software Engineering, 2000.

[10] D Jurafsky. (2016, December) Naive Bayes, From Languages to Information, Stanford University. [Online].

[11] Huliman, Analisis Akurasi Algoritma Pohon Keputusan dan k-Nearest Neighbor (k-NN),Tesis, Universitas Sumatera Utara, 2013

[12] Amazon SageMaker-Developer Guidehttps://docs.aws.amazon.com/sagemaker/latest/dg/how-pca-works.html .10/11/2018 -3:50 PM.

[13] M.H. Dunham, DATA MINING Introductory and Advanced Topics, USA: Prentice Hall, 2003.