A Review on Test Automation for Test Cases Generation using NLP Techniques

Srinivas Perala*
Department of Electronics, Lovely Professional University, Phagwara, Punjab, India.

Dr. Ajay Roy
Department of Electronics, Lovely Professional University, Phagwara, Punjab, India.
* Corresponding author. E-mail: srinivaslabview@gmail.com

Article History: Received: 10 November 2020; Revised 12 January 2021; Accepted: 27 January 2021; Published online: 5 April 2021

Abstract: In the process of product development, stakeholders and top management summarize the concept and document the requirements in natural language. These ideas and descriptions documented as software requirements by the technical department. Developers develop software following this software requirement document. For testing this developed software, they derive test cases from natural language requirements and then do the testing process to find the bugs. This process involves understanding requirements and derives test cases that are used to understand by developers and testers. Due to increasing the advanced features, deriving the test cases is monotonous and takes more time. This research article shows a method to automate this process which is deriving test cases from requirements using NLP algorithms. This approach useful to reduce the time and cost of software development.

Keywords: Natural language processing (NLP), Test automation, software cost reduction (SCR), support vector machines (SVM).

1. Introduction

Thinking ahead, Data Science usage progressively advanced without wanting to change the existing hardware. Also, Artificial intelligence (AI) Algorithms can apply to the right processor and operating system computers. Natural Language Processing (NLP) is one of the subsets of AI that add intelligent feature like the ability to read and understand like humans. The NLP-based application has already been used and applies to medical, media, financial, human resources, and more. Most of the data is found in text format which includes symbols and few other things. Since some data contains rich information it is a need to extract data and use it accordingly. To complete it, this requires that NLP can analyses our data and perform functions such as emotional analysis, cognitive facilitator, time filtering, targeting misinformation, and real-time language translation. To achieve the best results the NLP system must have a high-level algorithm which can enhance a good understanding of text and symbol. There are many methods available to help the system to understand text and symbols. Text separation, semantic vector, word embedding, possible language model, sequential labelling, and word editing.

Text classification processes the text and divides it into similar groups of words. Text classification algorithms process the input text then assign a set of pre-defined tags and symbols. NLP is used for sentiment analysis, topic detection, and language detection. There are primarily three text classification methods rules-based, machine system, and hybrid.

In the rule-based method, texts divided into a labelled group using a set of handicraft linguistic rules. Those handicraft linguistic rules contain users to define a list of words that are categorized by clusters. For example, words like Narender Modi and Rahul Gandhi would be considered into politics group. People like Sachin and dhoni would be considered into the sports group. Machine-based classifier learns to make a classification based on historical observation from the data sets. User data is prelabelled as tarin and test data. It collects the classification strategy from the previous inputs and learns continuously. The third method to text classification is

the Hybrid Approach. Hybrid based method usage of the rule-based system to create a tag and use machine learning to train the system and create a rule.

NLP is a subset of artificial intelligence that deals with machines and human language. In particular, how to program machines to understand and process the natural language data. In the product life cycle development, mostly the test cases developed by software developers using software requirement document. This work involves understanding natural language and must derive a logical requirement to test each feature of the software.

2. Literature Review

The complexity of software installed in sensitive security environments, e.g., automotive and avionics, has grown significantly over the years. In such cases, soft material testing plays an important role in ensuring that the system meets all operational requirements. Such a system test task is often referred to as an acceptance test. Contrary to validation, which aims to detect bugs, acceptance testing is a confirmation function performed at the end of a life of developmental life to show compliance with requirements. Acceptance testing is mandatory and regulated by international standards. For example, the balance between the requirements and conditions of acceptance testing is enforced by the standards of embedded security systems. Exploratory cases in particular are found in the occupational safety requirements expressed in the Nature language. Sometimes experimental cases are also found using rational mathematical models. Using a System Model Case System, Acceptance Tests Generation (UMTG) certifies the implementation of practical, systematic test cases, which accepts acceptance from the definition of needs in the native language, and reduce the efforts required to generate test cases and ensure coverage requirements. Several methods can produce test cases including input data right from NLP requirements specification. Token-making, branded business acknowledgement, and POS tagging are well-known in the software engineering community because they have been adopted by many methods including NLP. Business recognition and POS marking are often used in the software as they have been implemented by many methods including NLP techniques. Transforming needs into Work drawings, Statecharts and sequence diagrams, UML-supported models help generate test cases. Linear Discriminant Analysis is a powerful technique for data classification and reduction. It helpful where the within-class frequencies are unequal and their performances have been estimated on randomly produced test data.

3. Methodology

The proposed method is shown in fig.1 and it takes input as a requirement in the English language and gives the output as test cases.

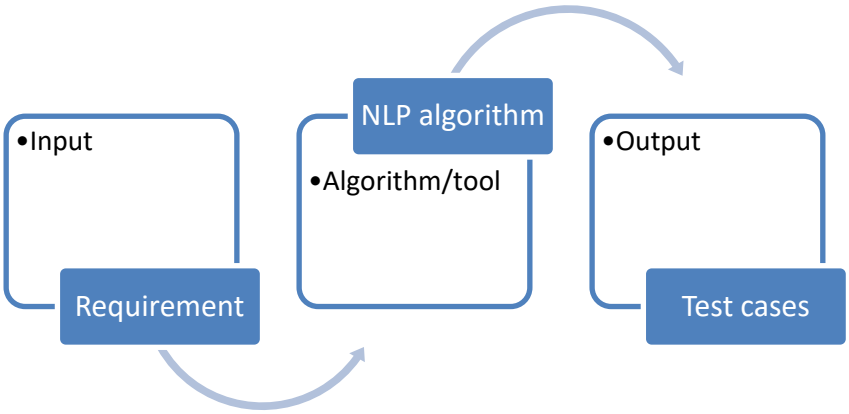

Fig. 1. Block diagram of the proposed system

Software Requirement: After Oven Power on, If the Oven door open, Oven Light must turn on and must record oven light on/off count in OvenLightOpenCloseCount and Oven Door open/close count must record in OvenDoorOpenCloseCount.

The above requirement processed by the NLP algorithm and will give the following output.

Action	Function/Service Type	Parameter	Value
Setup			
Set	Function	Test Unit	Power ON
Main Steps			
Read	ECU Variable	OvenDoorOpenCloseCount	X_Count
Read	ECU Variable	OvenLightOnOffCount	Y_Count
Set	ECU Variable	Oven Door	Open
Check	ECU Variable	Oven Light	ON
Set	ECU Variable	Oven Door	Close
Check	ECU Variable	Oven Light	OFF
Check	ECU Variable	OvenDoorOpenCloseCount	X_Count + 1
Check	ECU Variable	OvenLightOnOffCount	Y_Count + 1
Clean-up			
Set	Function	Test Unit	Power OFF

Table 1. System generated test case

We used the Naive Bayes, support vector machines (SVM) NLP algorithms for text classification from the requirement. This involves six levels as follows.

- In level 1, Defined required labels and constructed libraries.
- In level 2, Added the corpus can easily read data set using pandas read functions.
- In level 3, Data preprocessing implemented using Tokenization and word stemming.
- In level 4, label encoding for the target variable.
- In level 5, For Word Vectorization, used Term Frequency -Inverse Document Frequency (TF-IDF) which are the components of the resulting scores assigned to each word.
- In level 6, Used Naive Bayes Classifier Algorithm to predict the output.

4. Conclusion

The proposed method is very useful in the embedded software testing process. Also, developers and test engineers have very clear tracking of software requirements. Of course, there is some limitation in the automation. The maximum complexity goes to 10 to 20 % project and this can be handled manually. The complexity of manual work can automate with help of the intelligent logics and learning input data.

.

References

[1]. E. Cambria and B. White, "Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]," in IEEE Computational Intelligence Magazine, vol. 9, no. 2, pp. 48-57, May 2014, doi: 10.1109/MCI.2014.2307227.

[2]. J. Ding and S. Li, "A Low-Latency and Low-Cost Montgomery Modular Multiplier Based on NLP Multiplication," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 7, pp. 1319-1323, July 2020, doi: 10.1109/TCSII.2019.2932328.

[3]. D. Falessi and G. Cantone, "The Effort Savings from Using NLP to Classify Equivalent Requirements," in IEEE Software, vol. 36, no. 1, pp. 48-55, Jan.-Feb. 2019, doi: 10.1109/MS.2018.2874620.

[4]. P. García-Plaza, V. Fresno, R. M. Unanue and A. Zubiaga, "Using Fuzzy Logic to Leverage HTML Markup for Web Page Representation," in IEEE Transactions on Fuzzy Systems, vol. 25, no. 4, pp. 919-933, Aug. 2017, doi: 10.1109/TFUZZ.2016.2586971.

[5]. Arora, M. Sabetzadeh, L. Briand and F. Zimmer, "Automated Checking of Conformance to Requirements Templates Using Natural Language Processing," in IEEE Transactions on Software Engineering, vol. 41, no. 10, pp. 944-968, 1 Oct. 2015, doi: 10.1109/TSE.2015.2428709.

[6]. S. Gehrmann, H. Strobelt, R. Krüger, H. Pfister and A. M. Rush, "Visual Interaction with Deep Learning Models through Collaborative Semantic Inference," in IEEE Transactions on Visualization and Computer Graphics, vol. 26, no. 1, pp. 884-894, Jan. 2020, doi: 10.1109/TVCG.2019.2934595.

[7]. Kumar, P. K. Verma, S. Perala and P. R. Chadha, "Automatic attendance system by visual programming language LabVIEW," 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), Delhi, India, 2016, pp. 1-5, doi: 10.1109/ICPEICES.2016.7853192.

[8]. Bleaman, I.L. Predicate fronting in Yiddish and conditions on multiple copy Spell-Out. Nat Lang Linguist Theory (2021). https://doi.org/10.1007/s11049-021-09512-3

[9]. Gao, G., Wang, H. & Wüthrich, M.V. Boosting Poisson regression models with telematics car driving data. Mach Learn (2021). https://doi.org/10.1007/s10994-021-05957-0