Automated Test Script Generation from Natural Language Query

Deepamala.N, Tushar Kanakagiri, Shreyas Raghunath, Sughosh Kaushik, Dr.Shobha G, Ankit Singh, Deepak jha

Abstract*: *Development and Testing is a very important part of any product development cycle. There exist numerous modules that need to be tested in a product or software after each build. Addition, modification or deletion of new procedures requires thorough testing of the complete product. Test scripts are generated for the ease of testing. It is observed that most of the procedures in test scripts are repeated. Converting natural language query into test scripts reduces the effort of the test engineer by finding relevant procedures in already existing database. The proposed system accepts a natural language query and converts the query into an executable test code using various NLP techniques. This paper explain two methods that are used to generate test script from Natural language query.

***Index Terms*: *Natural Language query; Test Script generation; Intent Recognition*.**

I. INTRODUCTION

Product Development Lifecycle (PDL) involves idea, research, development, testing and analysis phases. Testing is a very important part where the product developed is tested for completeness, performance and to find bugs. Testing is carried out after every build and also after every minor modification in the code. Testing involves test scripts written using scripting languages like Javascript, perl, python, Jscript etc. The test scripts are stored in a database and are run to test the complete product. It has been observed that most of the testing scripts are duplicate codes. It is difficult to find relevant test scripts from the database.

The current paper discusses a mechanism where a Natural Language query interface is given for the test engineer to enter the test requirement in the form of a Natural language sentence. The sentence is parsed to find the intent using intent recognition mechanism. Based on the intent, the relevant test scripts are identified and an executable test script is generated. This test script can be run to get the testing results.

II. EXISTING SYSTEM

Revised Manuscript Received on May 5, 2019

Dr.Deepamala.N· Department of Computer Science and Engineering, R.V.College of Engineering, Bangalore, India
Tushar Kanakagiri· Department of Computer Science and Engineering, R.V.College of Engineering, Bangalore, India
Shreyas Raghunath· Department of Computer Science and Engineering, R.V.College of Engineering, Bangalore, India
Sughosh Kaushik· Department of Computer Science and Engineering, R.V.College of Engineering, Bangalore, India
Dr.Shobha G· Department of Computer Science and Engineering, R.V.College of Engineering, Bangalore, India
Ankit Singh Citrix Systems, Bangalore, India
Deepak jha Citrix Systems, Bangalore, India

In this section, several literatures of Parser and Comment extractor, Part of Speech Tagging, Language Understanding and Intelligent Service are discussed. Also for each of the above topics, the advantages, disadvantages and the challenges encountered are discussed.

A. Parser and Comment Extractor

Probabilistic parsing is one of the first techniques to make a breakthrough in extracting the keywords from the natural language inputs. While the method is not much used today, the variants of the probabilistic parsing are used extensively [1]. While the method of probabilistic

Parsing works perfectly for some normal statements, it is not effective for statements involving technical words, for example quicksort. Unlike the normal probability space, for technical words there is a need for sophisticated ones.

One can improve the normal probabilistic parsing by introducing random variables instead of using plain probability spaces. In addition to the probability function, which is from the power set of tokens to the interval [0,1], additional functions from the sample space to measurable sets are constructed. This additional function will take of the technical keywords by developing hierarchy among the technical words. The introduction of random variables also simplify the model. Most of the dependency parsers classify based on millions of sparse features. However, these features generalize poorly and are also computationally expensive [2].

Dependency parser using neural networks proposed a neural network parser using transition based approaches. The neural network learns word representations, POS tags. The first layer computes the word embedding and the second layer does the POS tagging. Since the words are embedded in the first layer, the sparsity is removed by the output layer [3]. Deep learning techniques for syntactic parsing were developed where they propagate information through the tree using recursive compositional procedure, proceeding step by step in a greedy manner. A neural network sliding window is used to separate the input sequences. The advantage of this procedure is that it is dynamically more stable compared to other methods, and can consider the data batch wise [4]. In the work on neural parsing used the modified approach to the graph based approach for parsing. The works aims at building a larger neural network. But since it more prone to overfitting, a larger regularization is used to avoid overfitting. Instead of using traditional multilayer perceptron based attention mechanism, affine layer classifiers with bi affine ones are used. First the dimensionality is reduced to

remove unnecessary features and then it is trained like a normal network. The advantage of this approach is that it is computationally effective [5].

B. Parts of Speech Tagging

Christopher Manning in his paper 'A maximum entropy model for Part of Speech Tagging' (2011), discusses training a corpus for Part of Speech Tagging using maximum entropy model and contextual features. The corpus has an accuracy of 96.5% on the test data. A maximum entropy model is well suited for diverse forms of contextual information in principled manner. It doesn't impose any distributional assumptions on the training data [6].

Part of speech tagging can be done using a tagger network, which is essentially a feedforward neural network and a lexicon. The tagger network contains many layers of simple perceptron, also called as a multilayer perceptron. Each unit in the output layer of the multilayer perceptron denotes one of the possible tags in the target tags. The sentence is first converted into a form that the neural network understands, and is then passed onto the input layer. Each node in the input layer first computes a pre-activation and then nonlinearity is applied to compute the neuron activations. Such activations are passed onto the next layers, which in turn compute more abstract features [7]. Recurrent neural network was used to drastically improve the accuracy of the part of speech tagging. Simple recurrent neural network, SRN, was used to compute a new state based on the previous and current state using a single layer perceptron. Simple recurrent neural networks are trained using two phases. First, the text is ambiguously tagged using Analyser. In the second phase, each ambiguous tag is compared to all possible tags, and using F-score ambiguity is eliminated [8].

C. Language Understanding Intelligent Service

An animation system that generates an animation from natural language texts such as movie scripts or stories was developed. The system does semantic analysis to find the motion clips based on verbs [9]. Invention relating to the creation of computer database systems and the querying of data contained therein was done in [10]. The steps involved generation of fact tree based on natural query, check query for semantic correctness and generate query for the database. Conversion of business rules written in natural language to set of executable models as UML, SQL etc. was carried out in [11].

Intent recognition involves finding the intent of the sentence along with arguments. The artificial legs have three locomotion modes walking, stair accent and stair decent. The locomotion intent of the subject is identified using SVM [12]. [13] Discuss the identification of human intent in context of human-robot interaction. Hidden Markov Models (HMMs) have been used to classifying limited numbers of gestural patterns. They take the context into account. Relating the context and history of interaction to the kinematics is a key point for recognizing human gestures in HRI. A comparative study of cloud platforms to develop Chabot is discussed. User input is processed through two modules: intent classification and entity recognition. External APIs and algorithms are used to generate response [14].

Language Understanding Intelligent Service (LUIS) helps to create models for the applications to better understand the intents or entities. LUIS helps developers to build applications that can understand human language and react to users accordingly. It supports various languages such as English, German, Italian and French. The advantage of LUIS is that one need not have to worry about explicit tokenization [15].

III. PROPOSED SYSTEM

The system was implemented using two methods. One using term frequency method and the other use LUIS API.

A. Method 1: Using term frequency method

The architecture of the system is divided on the basis of requirement in the order of execution. This allows the architecture to assume a highly modular structure consisting of extraction, pre-processing, building function signatures, mapping and parameter extraction.

The first main module as shown in Figure 1 of the architecture comprises of a parser that is used to parse a variety of functions and identify comments, both single and multiline. It also extracts the same and utilizes it to populate a database of functions.

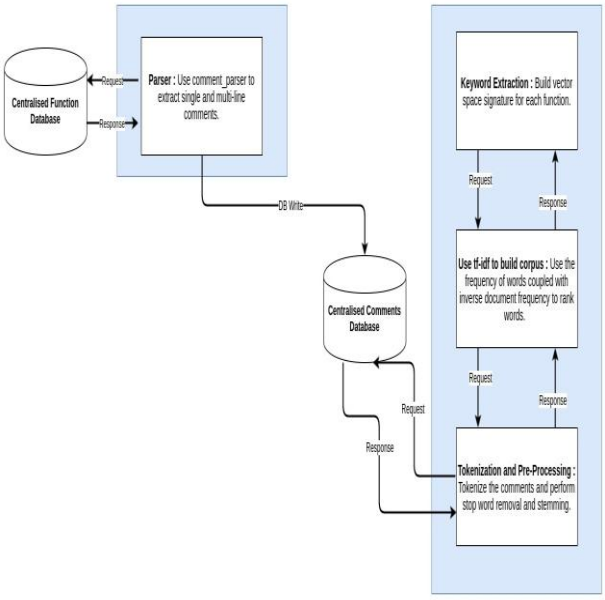

Fig. 1: High level system architecture - Parsing and Keyword Extraction

The next section of the architecture focuses on extracting keywords and uses various models to help build a corpus of words that help in pre-processing. This module also interacts with the database to access the previously populated function list. This module ends with building a vector signature for each function in the database, which is made as unique as possible. Important keywords in text is determined by TF-IDF method. It picks the important words in a document, where

$$tfidf_{i,j} = tf_{i,j} X \log(\frac{N}{df_i})$$

$tf_{i,j}$= total number of occurrences of i in j

df_i= total number of documents containing i
N= total number of documents

The next part of the architecture is more user/client facing in nature as shown in Figure 2. It enables the user to pass a query in the form of Natural language. The above mentioned pre-processing is also performed on the user input. A similar vector signature is built.

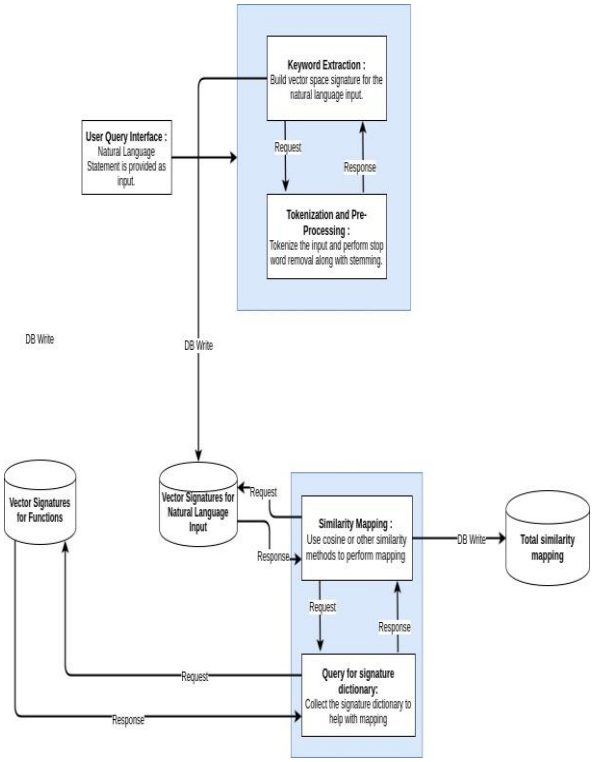

Fig. 2: High level System Architecture - Vector signature and Function mapping

The next module of the architecture is the crucial part of the function mapping that utilizes various models to map the vector signatures of input to the existing vector signature database.

B. Method 2: Using LUIS API

In another method, LUIS APIs are used to fetch the intent of the given sentence with the entities. Language Understanding Intelligent Service (LUIS) [15] helps to create models for the applications to better understand the intents or entities. LUIS is a toolset and runtime application for the generation and execution of simple language understanding models. It has two main components, Entity Extraction, to extract an expected keyword, and Intent Resolution to determine whether a given word matches a prespecified intent. There are mainly two level interfaces, the Registrar, which provides APIs to register and remove models, and the run time core which provides language understanding facilities.

The Registrar surfaces two simple APIs, but internally contains logic to copy model data into the internal database, serialization and deserialization logic, and multithreaded initialization of Executors.

The Executor isolates the LUIS API boundary as taken from the server team so that the LUIS codebase did not require large changes when being moved from the server to on-device. It directly wraps the LUIS APIs for Entity Extraction and Logistic Regression and ensures the data provided to the LUIS APIs are in the format expected [15].

The user given query is sent to LUIS using APIs to get the intent and its entities. This is mapped to the keywords obtained from the test functions. The matched test scripts are converted to executable code as shown in figure 3.

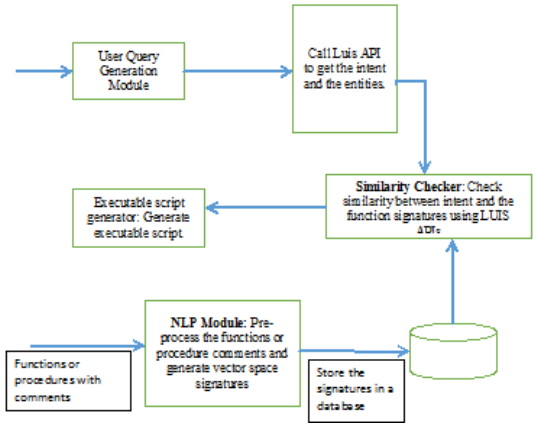

Fig. 3: High level System Architecture – LUIS API

IV. RESULTS AND ANALYSIS

The interface (GUI) built allows the user to easily enter natural language input. The text box serves as a medium to enter the input. In Figure 4, 5, 6 multiple kinds of input is provided. This is an intuitive interface for programmers. In Figure 4 the user enters the command. This query sends a post request to the server. The flask server responds by performing the required translation and writing the function call to the source file.

Fig. 4: Entering first function and clicking on ADD FUNCTION button

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Fig. 5: Entering second function and clicking on ADD FUNCTION button

Fig. 6: Entering third function and clicking on ADD FUNTION button

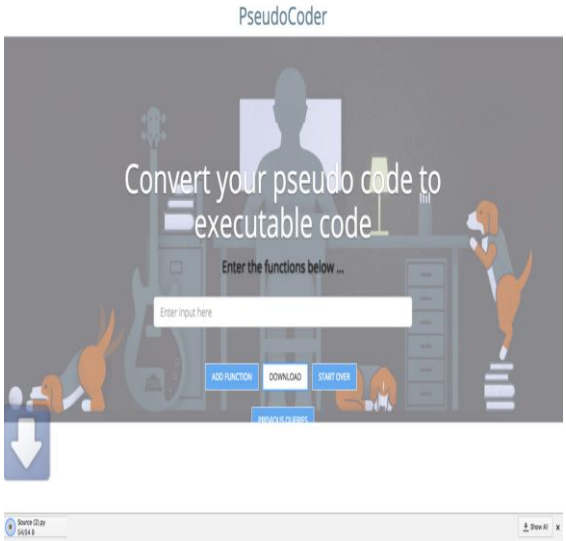

Fig. 7: Script getting downloaded

In figure 7 the user downloads the final script that is generated. This can be accessed from the downloads folder. This is ready to execute, "source file" is the final converted script.

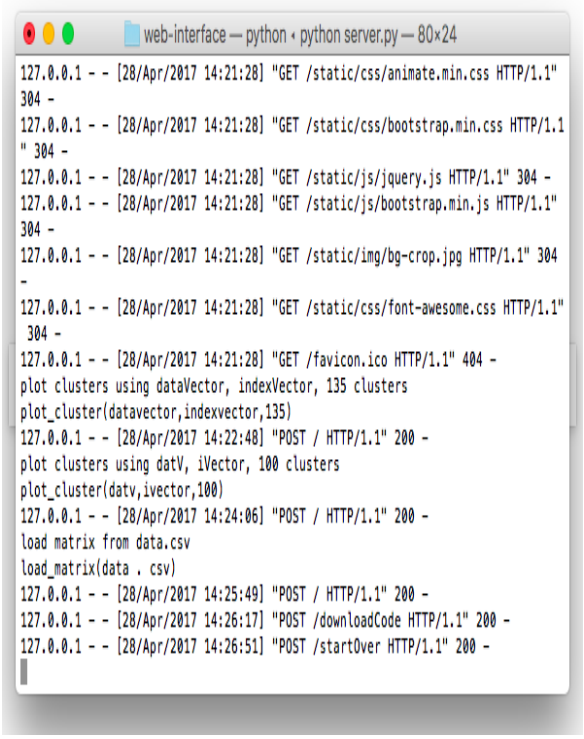

Fig. 8: Server log in terminal

In figure 8, the server log for the application can be viewed in terminal, this helps in debugging the code as needed, when errors arise. The server is a flask server that allows an easy debugger mode. The server log also allows us to keep track of the various resources that the interface is utilizing.

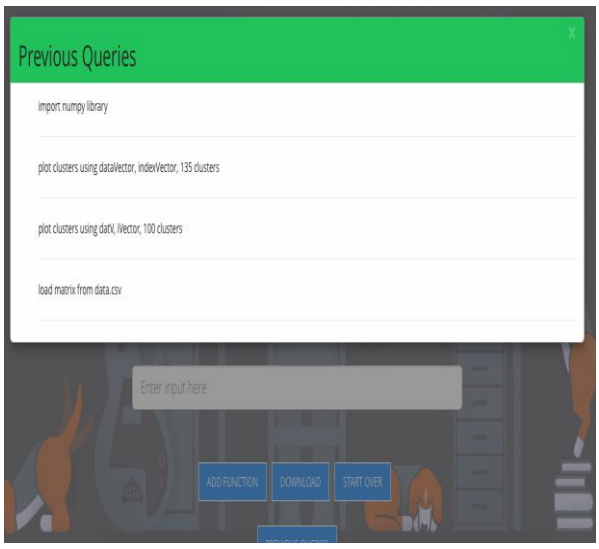

Fig. 9: List of queries that have been entered

Often the user enters multiple queries, which introduces the difficulty of remembering the past queries in large scripts. The feature shown in figure 9 for viewing the previous queries helps the programmer maintain context while writing the script.

The test was performed on company provided test cases. It was observed that, the first method where the tf-idf approach is used the result was inaccurate. As the corpus for technical words is unavailable and in most cases this method classifies the technical terms which are required as stop words and eliminates them.

The second approach is using the Microsoft Cognitive Services. The Luis.ai library is a Language Understanding Intelligent Service, a toolset and runtime for generation and execution of simple language understanding models [15]. A LUIS api maps to a set of binary intent classifiers and entity extractors. This method yielded the best results.

V. CONCLUSION

Generating test scripts is a time consuming task and many times it's repetitive. The tool converts a Natural language query into executable test script. The above tool uses TF-IDF method and LUIS built-in APIs to get the intent of the query. It has been observed that intent recognition works better than the frequency method. The tool not only helps reduction in redundant code but also makes the testing process faster.

REFERENCES

1. Mark Nederhof, Giorgio Satta. "Probabilistic Parsing", International Journal on Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 8, August 2012, pp 45-49.
2. Barr Hirrel. "Probabilistic Parsing with Random Variables and Probability Spaces", International Journal of Computer Sciences and Statistics, Vol 11, Issue 2, March 2014, pp 16-20.
3. William Chen, Sebastian Chu. "Dependency Parser using Neural Networks", Journal of Machine Learning Research, Vol 25, January 2015, pp 7-13.
4. McCulloch, Rosenblatt. "Deep Learning Techniques for Syntactic Parsing", International Conference on Machine Learning, Vol 13. Issue 3. April 2013, pp 25-31.
5. Alexander Grothendieck. "Attention for Neural Parsing", Neural Computation Society, Vol 8, Issue 2, July 2009, pp 53-60.
6. Christopher Manning. "A Maximum Entropy Model for Part of Speech Tagging", Journal of the ACM, Vol 16, Issue 4, December 2011, pp 267-283.
7. Hugo Larochelle, Pedro Domingos. "Part of Speech Tagging using Lexicons and Feed Forward Neural Networks", Engineering Applications of Artificial Intelligence, Vol 32, Issue 3, August 2012, pp 541-553.
8. Yoshua Bengio, Michael Forcada, "Improving Part of Speech Tagging using Recurrent Neural Networks", Neural Information Proceeding Systems, Vol 14, December- 2016, pp 78-97.
9. Oshita, M. (2010). Generating animation from natural language texts and semantic analysis for motion search and scheduling. The Visual Computer, 26(5),pp 339-352.
10. Harding, James A., and Jonathan I. McCormack. "Method and apparatus for the modeling and query of database structures using natural language-like constructs." U.S. Patent No. 5, 27 Feb. 1996, pp 495-604.
11. Zhang, Fan, et al. "Real-time implementation of an intent recognition system for artificial legs." Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE. IEEE, 2011.
12. Nehaniv, Chrystopher L., et al. "A methodological approach relating the classification of gesture to identification of human intent in the context of human-robot interaction." Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on. IEEE, 2005.
13. Patil, Amit, K. Marimuthu, and R. Niranchana. "Comparative study of cloud platforms to develop a Chatbot." International Journal of Engineering & Technology, 6.3, 2017, pp 57-61.
14. Croft, D. K. E. A. "Estimating intent for human-robot interaction." IEEE International Conference on Advanced Robotics. 2003.
15. Jason Williams, Eslam Kamal. "Fast and easy language understanding for dialog systems with Microsoft Language Understanding Intelligent Service (LUIS)", Microsoft Research, February 2016, pp 70-98.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication