An investigation of the paper "NL2Type: Inferring JavaScript Function Types from Natural Language Information"

Subtitle as needed ***(paper subtitle)***

Safa Keskin
Istanbul,
Turkey,
keskinsaf@itu.edu.tr

***Abstract*— JavaScript is a dynamically typed scripting language that is mainly developed for making DOM (Document Object Model) manipulation possible in the browsers. Dynamically typed languages help the newcomers to obtain results in short time, thanks to the automatic type casting that are applied during transpiling, but that also makes it hard to maintain big projects. Types are really important to give fundamental information about methods or properties. So, the information provided by type signatures are not visible for dynamically typed languages, but with the fast evolution of the computational power, extracting type signatures using the natural language processing tools is possible. The paper discussed through this investigation paper mainly uses LSTM architecture to obtain type signatures from human readable comments. Malik and his friends proposed that by analyzing human readable sentences with machine learning algorithms, types can be automatically inferred for JavaScript language.**

Keywords: JavaScript; type; LTSM; NLP; Natural Language Processing;

I. INTRODUCTION

Dynamically typed languages are used widely in software projects. JavaScript is one of them, which is introduced with the purpose of making DOM manipulation possible in browsers. With the carrying browser engine out of the browser with Node.js, it became possible to use JavaScript independent from the browser. That made JavaScript a language that can be used in any parts of a project. But due to its dynamically typed nature, it is really hard to obtain enough information about flow of a function just by considering function name and arguments. Malik and his friends proposed a tool named as NL2Type which analyses the comments of each function including function and argument properties. Their solution produces very high accuracy with provided data and according to my tests, the tool works well, too.

II. PREPARATION

A. Background

1) Long-Short Term Memory

LSTM is a neural network structure that is basically designed for destroying the vanishing problem in recurrent neural networks. LSTM structure is generally used in semantic analysis problems in order to benefit from its ability to use information collected through training. Because words are basically sequences that are related to each other, and the information can be extracted from that sequence with the LSTM.

B. Tools

1) JSDoc[1]

Documentation with comments is a common documentation style. JSDoc is a tool for that purpose in the JavaScript language. It has its own annotations, like @param for indication arguments and @return for indication return value descriptions.

2) Python NLTK[2]

Python Natural Language Tool-Kit library is a frequently used tool while working with human understandable data. It is used in many different applications including NLP applications and language related data clearance operations.

3) Keras[3]

Following its official definition, Keras is a deep learning API running on top of machine learning platform *TensorFlow*[4].

4) TensorFlow

TensorFlow is a platform for machine learning applications. It has thousands of modules that are provided as an open source project and actively maintained.

III. ALGORITHM

The human readable sentences are meaningful for humans but nothing more than a sequence of characters for a computer. So, a mapping from human languages to machine understanding should be performed to obtain meaningful results from an algorithm.

This work is based on an assumption related to the softwares: Comments include meaningful words and combinations of words that are used to the behavior of the method or characteristics of the arguments / return values. That is the general case for almost all projects. Writing comments to explain behavior of the method / class is an accepted behavior for the software development community although there are some arguments against that behavior. With the purpose of documenting code blocks with the comments, there are many tools in the literature for different languages. For example, JavaDoc[5] is used while writing comments for

documentation in Java projects, Godoc[6] is an accepted tool for the same purpose in Golang, and the same goes for JSDoc for the JavaScript language.

Developers write comments to explain behavior of code blocks using annotations like @param and @return in JSDoc. Following these annotations, types of the argument or definition of the behavior is included into comments. In order to convert that sequence of words, There are some operations that should be applied. For that purpose, writers of the paper used different tools in order. They used JSDoc itself to extract comments from files which includes both the method definitions and comments. After that, they wanted to convert obtained words to their first form if they are verbs, and remove unnecessary punctuations. They also aimed to remove the words that make sense for humans while reading, but make no sense for computers like "the", "a / a\n" and "and". At the end, they succeeded to obtain the pure forms of sentences that can be used while training the model. While processing the sentences in the mentioned direction, authors have used Python NLTK library which is de facto tool for such purposes in Python languages.

When the cleaning operation is completed, the data that will be trained is ready. With that data, by using the LSTM algorithm, really good results are obtained that are already mentioned in the paper. I will not dive into that results, because in this work I thought that I will present a report that is investigating the project. So, I will move on to the challenges I have faced during the investigation of the project.

IV. Challenges

Running the algorithm in a container using docker was easy. It was possible to run algorithm in a container just by pulling the image that authors named in their repo[5]. In my first try, I succeeded in running it as specified. But using source code was not so easy and I faced many challenges in order to test algorithm with my own test files.

Firstly, the repo was old and versions were not specified in the *requirements.txt* file. So, I have faced different problems. First problem version of the NLTK. The project probably was built with version 3.0, but current version is 3.5 and when I run the code as the authors specified, I got an invalid syntax error. That is not only related to the NLTK, but also related to the Python version. Support for Python 2.7 is ended after 2020, so it looks like NLTK is affected by that issue, too. Another problem was related to Keras and TensorFlow versions. All problems were related to the versioning. The tools were nott working consistently with each other. So, I have made changes in the requirements.txt and changed below packages version as mentioned in Table 1.

Fixing versioning took a long time, because errors were not descriptive and it was hard to find reasoning behind them. Also, Python itself provides some issues with versioning. It doesn't warn you when you have misconfigurations. For the dynamically typed languages, predicting error before it occurs is really hard. This can be the issue for another project.

Name	Version
tensorflow	1.14.0
keras	2.1.3
numpy	1.16.1
google-api-python-client	1.7.11
google-auth-oauthlib	0.4.1
nltk	3.0

There was another big problem I have faced. NLTK was a tool that highly depends on the language datasets. The project was also using NLTK, and if you wish to run the project on your local machine, you have to download *"maxent_treebank_pos_tagger"* in your local machine. It was hard to find the reason behind that error, because the program was only writing a log like *"english.pickle resource in NLTK is not found"*. For a person who is not familiar with NLTK, that error does not mean much. But with some research, I have found the reason as a response to one question asked at stackoverflow.com[7].

When everything is downloaded and versioning problems are resolved, I succeeded to run the algorithm in my local machine and applied some tests. According to my observations, authors were correct in their tests.

V. Conclusion

I misunderstood the purpose of the project and tried to make a good presentation about the tool I have chosen and tested. The project was mainly related to developing something new using some previous works, or even from scratch. Sorry for that. I have forked GitHub repository of the project I have worked on, and added my changes to there. I have also added my presentation there.

Although I misunderstood and cannot present you a good report that has meaningful things to you, I hope my GitHub repository[8] helps somebody who tries to run that algorithm from scratch, too.

References

[1] https://jsdoc.app/
[2] https://www.nltk.org/
[3] https://keras.io/
[4] https://www.tensorflow.org/
[5] https://www.oracle.com/java/technologies/javase/javadoc-tool.html#javadocdocuments
[6] https://blog.golang.org/godoc
[7] https://stackoverflow.com/q/4867197/6013366
[8] https://github.com/keskinsaf/NL2Type