Muhammad Haseeb 20100192
Aafaq Sabir 20100196

Using Natural Language Processing Principles for Categorizing Modern Webpages JavaScript code

1 Overview of the problem

Modern webpages have a lot of JS scripts embedded in them and a significant proportion of those scripts is not critical for the functionality of the webpages. Therefore, if we could identify those non-critical scripts then we can take various actions for simplifying the webpages thus reducing CPU usage and network bandwidth - both of which are crucial in the context of developing countries. In this project, we aim to use Natural Language Processing techniques or develop a technique based on NLP principles for classifying different JS scripts into various categories e.g. ads, marketing, tracking, etc.

2 Methodology

Below we have explained our methodology. Specifically, we have first described our preprocessing techniques and then we have described our modeling techniques.

2.1 Preprocessing

Our data consists of 117,000 JS scripts which we have crawled by developing a selenium based web JS crawler. For being able to apply different NLP models, we preprocessed the data by finding out all the DOM API calls present within the scripts. After extracting the API calls, we used n-grams of length 2, 3, and 4. We performed a study for finding out the optimal number of n in n-grams where we found the spatial difference between API calls present in the scripts and found out that usually, a function has on-average 4 API calls and then there is no API call until the next function or method.

After this preprocessing, we had scripts represented as API calls and their n-grams, and our vocabulary consisted of 2600 features. Then we reduced our vocabulary size by finding out the correlation between all the features and removed the features having more than 80% correlation and we ended up with 500 final features.

2.3 Modeling

For a pilot study, we have used TF-IDF vectorization with Random Forest classification and achieved 60% accuracy. Next, we are building a model like word2vec for JS scripts. We have found code2vec by Facebook which is a special embeddings technique developed on principles of word2vec but uses Abstract Syntax Tree's path contexts instead of keywords found in text/script. So we are going to use code2vec for our classification task and try to improve the accuracy further than the 60% which we achieved in our pilot study. Afterward, we are going to

use Long Short-Term Memory networks for our classification problem contingent on the available time after completing the code2vec part.

TL;DR we are going to use below techniques and find out the impact on our classification accuracy of JS script:

1. **Random Forest Classification**
2. **Code2Vec (a Word2Vec model for code snippets)**
3. **LSTM networks**

Currently, we are on the second point (code2vec) and the problem we are facing is that Facebook developed the support for C# and Java language only while our data is in JS language, so we are making code2vec compatible with JS language which would be itself a significant contribution.