PROGRESS REPORT

Deka Auliya Akbar
U1323056K
DEKA0001@e.ntu.edu.sg

Peter
U1320143C
PETE0013@e.ntu.edu.sg

Stefan Artaputra Indriawan
U1320079K
SINDRIAW001@e.ntu.edu.sg

Stefan Setyadi
U1420118A
STEF0019@e.ntu.edu.sg

Zillion Govin
U1320174E
ZILL0001@e.ntu.edu.sg

ABSTRACT

In this report, we will describe our project plan and the project progress that we have made for the Natural Language Processing (NLP) project. The objective of this project is to get a hands on experience with the NLP tools by developing an application to recognize the APIs mentioned by users on Stack Overflow.

PROJECT PLAN

Below is the planned schedule and timeline for the project activities that we have done and planned to do for the next following weeks.

Table 1. Project Schedule

WEEK	ACTIVITY
5	Project Requirements
6	Methodologies and Dataset Collection
7	Data Preprocessing and Summary Statistics
8	Dataset Analysis, Selecting Data for Training, and POS Tagging
9	API Mention Annotation and Recognition*
10	Evaluation and Error Analysis*
11	Application*

*still under progress / future plans

1. PROJECT REQUIREMENTS

We tried to extract information from the assignment guideline, derive the requirements and understand the project expectations. We also consider the tools and technologies that we are going to use for the project.

2. METHODOLOGIES

We divided the problem into sub problems, first we perform dataset collection, then we preprocess the collected data and derive some statistics. Next, we define API Mention and perform some analysis. After that, we need to consider the tools for Named Entity Recognition (NER) and train a classifier to recognize API mention in our data. Last, we will evaluate and perform an error analysis on the trained NER Classifier, and also derive the possible applications for recognizing API mention.

2.1 TOOLS AND LIBRARIES

a. Python 2.7, Natural Language Toolkit (NLTK) library, Requests, BeautifulSoup
b. HTML and JavaScript
c. Stack Exchange API
d. Java 8, Stanford NER classifiers for helping with the text processing.
e. Brat annotation tool

3. DATASET COLLECTION

Our datasets include top voted 100 StackOverflow question posts tagged with Java collected together with their answer posts collected from 6 months' period between 2011-2015. This process results in obtaining 10 JSON responses of 8244 posts consisting of 1000 question posts and 7244 answer posts. The data is obtained by performing a REST API request to Stack Exchange API (https://api.stackexchange.com/). The collected data is then stored as a collection of JSON files.

4. DATA PREPROCESSING AND SUMMARY STATISTICS

Since StackOverflow is a web service, the dataset that we collect still contains HTML tags, therefore we first have to perform preprocessing. The preprocessing takes the input of raw JSON data, and then write the preprocessed data that has been cleaned and tokenized into a new JSON file.

For preprocessing, we used Beautiful Soup 4, a python library for cleaning html tags and NLTK, another python library for tokenizing the words. We decided to preserved some of the HTML tags, such as <p>, to indicate that the start of the sentence. We decided to exclude the content inside <pre> and <blockquote> tag because they mostly contains code, not sentences. We did not remove the tag but we replace them with a token to help indicate a part of the sentence, one reason is because not all sentences ended with a period ('.'). One of the example is: "Here is an example: <pre>…</pre>". In this case, our script would replace pre tag as a token "#pre" as a substitution to a period and discard the content inside <pre>.

For statistics, we use matplotlib.py plot to plot the histogram for number of answers per question from all of our dataset. We also counted the number of words in each post and total number of question and answer posts from all of the JSON files. Lastly, we performed stemming to obtain information about the most frequent words that appeared in the data set.

5. DATASET ANALYSIS

5.1 API Mention Definition

We limit our scope of focus to Java Programming Language. In other words, our model will only classify API mention based on Java. The definition of our API mention is fairly simple:

a. Class method. ex: Class.Method(), Class.Method
b. Instance method. ex: Object.Method(), Object.Method
c. Method call. ex: Method()

To sum it up, we consider every method call to be API mention. The Class can either be native Java class or user-defined class.

In our dataset training collection from StackOverflow, we select posts that meet our API mention requirements above. Additionally, those API mentions must fulfil these formatting requirements as well:

a. Inline
b. surrounded in <code></code> tag
c. not in a <pre></pre> tag
d. not in a <blockquote></blockquote> tag

We do not define any regulations for the number of parameters and non-existing parentheses as long as it is a valid method name in the given context.

5.2 TRAINING DATA COLLECTION

We selected our training data from the data collected with StackExchange API. Based on our API mention definition, we visualized all the JSON response in a more readable form using HTML and JavaScript. Then, we manually selected and extracted the post that contains the API mention with the help of a script.

5.3 POS Tagging

For this task, we randomly selected 10 questions from the whole dataset. By using NLTK Library, we apply word_tokenize function to the selected sentences line by line and saved the result in the form of the word and POS tag pairs.

6. API MENTION ANNOTATION

We plan to use IOB encoding to annotate whether a word or word phrase is an API mention. To do this, first we must have the preprocessed data tokenized. Then, we annotate the tokenized data with labels (I = inside, O = outside, B = beginning) in a tab separated format text file. We will also consider the option of using BRAT during the annotation process.

7. API MENTION RECOGNITION

To do the required API Mention recognition, which is a case of NER (Named Entity Recognition), we will create a classifier by using Stanford NER tool[1]. This tool provides an implementation of a linear chain Conditional Random Field (CRF). By using the annotated data with the IOB entity tag, we can train the model to recognize the structure of an API mention.

After the model is trained, we will test the model and refine as necessary on the feature and specifics of the training to achieve higher precision on the test data.

8. EVALUATION AND ERROR ANALYSIS

The evaluation for our API mention classifier is the F-score.

$$Accuracy = \frac{(\sum TruePositive + \sum TrueNegative)}{\sum TotalPopulation}$$

$$Precision = \frac{\sum TruePositive}{\sum TestOutcomePositive}$$

$$Recall = \frac{\sum TruePositive}{\sum ConditionPostive}$$

$$F1_{score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

To do this, we first split the data into training and test. After we train the model, we will then use the classifier to evaluate the classifier performance by using the test set.

[1] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. *Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics* (ACL 2005), pp. 363-370. http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf