Dependence of function name prediction accuracy on its length

Ilya Bykov

1 Motivation

The task of function name prediction is quite relevant because it can help reduce code-review and debugging time, it is useful for automatic documentation generation, semantic code search and so on.

When analyzing the factors, the relationship between the accuracy of function name prediction and the length of the function may play a great role. Solving this problem can significantly help to optimize the prediction model: if, suppose, the accuracy on short lengths is higher, we can divide the long code into blocks and train the model on smaller parts; if the accuracy is clustered for different code lengths, we can try to build several models for each class.

2 Hypothesis and experiment design

It is reasonable to assume that when the functions are too short, the model may lack enough information to predict the name. At the same time, if the function is long, it contains more information, but it may also contain a lot of extra noise, which will also make the prediction task tougher.
Therefore, my hypothesis is that the prediction accuracy for near-median function lengths will be higher for both shorter code blocks and longer ones.

In order to conduct an experiment to test the hypothesis, a predictive model is essential. Data preprocessing and tokenization, selection of a suitable dictionary for classification, and design of a predictive model are separate, time-consuming tasks, and to shorten the path to experimentation, let's take the project data as a basis: github.com/welel/function-name-prediction which is based on python functions.

The author solves the classification problem as follows: in the data preprocessing stage, an abstract syntax tree of the function body is fed, and to represent it as a vector, the author proposes to traverse it using the Depth-first search (DFS) method and feed it into the embedding layer. After the embedding layer, the author proposes several different models: convolutional neural network, LSTM or Bi-LSTM. The last one showed the highest accuracy (about 80%), so we use it. To read more about how the author preprocessed the functions and built the neural network, see in dataset preparation, model architecture.

3 Workflow and results

The notebook with my analysis you can find in this repository:
github.com/philurame/function-name-accuracy.

I followed the steps suggested in the work: I downloaded data with python functions from learnbigcode.github.io and then I processed the data and saved it to the files X_test_tokens.txt ,Y_test_tokens.txt, x_test.txt, y_test.txt. The first two datasets are tokenized input, the last two are vectorized. After training the Bi-LSTM model I saved the predictions in the file y_pred.pkl.

I decided to test the hypothesis with test data. Of which 30961 samples with 102 classes (different function names). I took the number of tokens in the ASD tree as the "length"of the function.

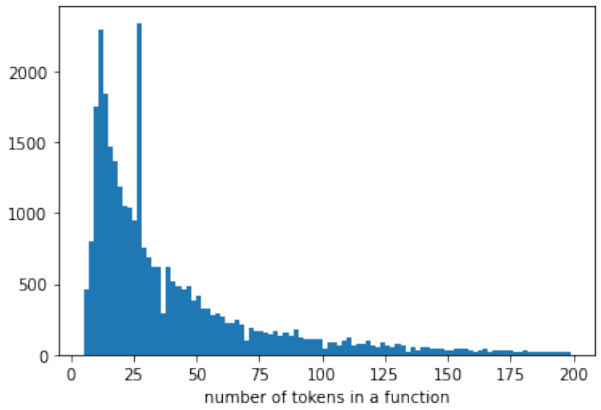

Рис. 1: Lengths distribution

Unfortunately, the data are very uneven, which may affect the statistical significance of the result. Now, to test the hypothesis let's split the whole sample into three roughly equal parts (using 0.32-quantile and 0.65-quantile) and to test differences in accuracy we should break each part into batches, calculate the accuracy on each and get ready samples.

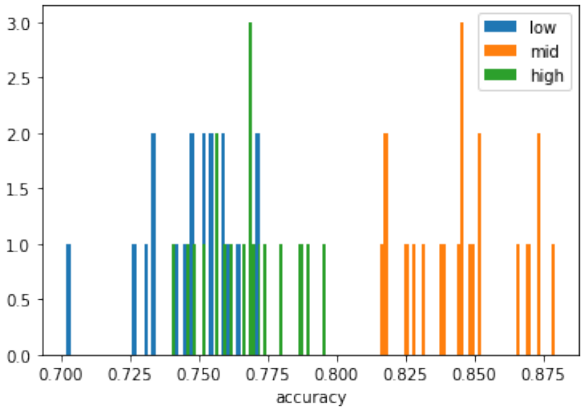

Рис. 2: Accuracy distribution over three length-groups

It is already obvious from chart that the model best guesses the names of near-median length functions. But for statistical validity we will conduct a test: since we know little about the distribution of data (or it is very difficult to calculate) we use the non-parametric Mann-Whitney U rank test, which showed that the p-value for all combinations of samples is significantly less than 0.05. This in particular suggests that the model is better at identifying names for a very long code than a very short code.

Thus it was found that for a given dataset and the chosen model, the greatest accuracy in predicting the function name is achieved at near-median length.

4 Threats to validity

As already mentioned, this study is rather incomplete: first, the study was conducted for a single model with Bi-LSTM. For a more general picture, it is worthwhile to conduct a similar experiment for different models (in particular, the convolutional and LSTM models proposed by the author) and then aggregate the results. Second, Python data was used, and generally speaking, the same functions in other programming languages will have a different structure (in terms of tree depth and with different tokens). Third, DFS traversal was used to straighten the ASD tree, and to see if it is statistically significant, study a class of ways to vectorize the tree (or in general also consider graph neural networks).