GenAI for Software Development: Assignment 1

Jonathan Hou

jhou03@wm.edu

Introduction

This assignment aims to create a Java code completion model. The selected type of model will be an N-gram model that predicts the next token in a sequence given a certain N number of tokens for context. It then calculates the probability of possible next tokens based on training data and chooses the token with the highest probability. The methods used to create this model will be mining Java code from GitHub repositories, cleaning and filtering the methods, tokenizing the code using Pygments, splitting the corpus using sci-kit-learn, and generating the model using Natural Language Toolkit MLE. The performance is measured by a calculated perplexity value. The source code for this assignment is available at

Implementation

GitHub Repository selection was done via the GitHub Search tool (https://seart-ghs.si.usi.ch/) to compile a list of well-written Java code. These repositories were filtered with the following: language=”Java”, minimum number of commits=25, maximum number of commits=2500, minimum number of contributors=15, minimum number of stars=200, last commit between 01/01/2024 to present day. This search yielded 1615 repositories, of which 100 were randomly selected. From these 100 selected repositories, I extracted 100,000 Java methods from 3990 unique classes.

These 100k Java methods were cleaned using the preprocessing script provided by the professor. After removing duplicate methods, filtering for only ASCII characters, and removing outlier methods based on method length, the resulting corpus is 37273 Java methods from 3748 unique classes.

I utilized the Pygments library and the JavaLexer class to tokenize the extracted Java methods. These methods were then exported to a .txt file which serves as the input to the N-gram model script.

The data set is split using sci-kit-learn, randomly selecting 80% of the methods as the training set, with the remaining 20% being split into 10% + 10% making up the evaluation and testing set.

I trained multiple models using N = 3, 5, and 7. The evaluation of the models was done by calculating perplexity, where lower values indicate better performance. The selected model trained from the mined corpus was n=7 with a perplexity of 1.319 on the evaluation set and 1.303 on the test set. The same was done using the teacher provided corpus, where the best model was also n=7 with a perplexity of 1.047 on the evaluation set and 1.056 on the test set.

Predictions were generated on a random sample of 100 methods and the results can be found stored in .json files.