GenAI for Software Development: Assignment 1

David Berger
dhberger@wm.edu

1 Introduction

The goal of this assignment is to train an N-gram recommender system that can probabilistically generate Java methods based on given code tokens. The data used to train the model comes from mined GitHub repositories using PyDriller, which are pre-processed and tokenized using Pygments. The nltk module is used to construct the N-grams used to generate code. The source code of the assignment can be found at https://github.com/dhbergerwm/CSCI420HW1.

2 Implementation

2.1 Dataset Preparation

GitHub Repository Selection: The GitHub Search tool (https://seart-ghs.si.usi.ch/) was used to mine repositories. The search was narrowed to Java repositories with between 500 and 2000 commits, at least 10000 code lines, and at least 1000 stars, allowing no forks. This yielded 604 valid repositories, although only 5 will be used. Using PyDriller, 228174 Java methods were extracted after traversing each of the repositories' commits.

Cleaning and Tokenization: The raw dataset is filled with redundant or outlier functions, so the methods were pruned using a set of pre-processing methods to remove duplicates, boilerplate methods, non-ASCII methods, overly long and short methods, and the comments for each method. After cleaning, 29373 methods remained. The Pygments Python package was used to tokenize the dataset methods. After tokenization, a vocabulary of 24206 tokens was produced, covering the training, validation, and testing sets.

Dataset Splitting: To split the dataset, the cleaned methods were randomly shuffled and an 80%/10%/10% split was made for training, validation, and testing respectively.

Model Training & Evaluation: Context sizes of n=3, n=5, and n=7 were tested to determine the best-performing N-gram model. The models were evaluated on the validation set based on perplexity, with the lowest perplexity representing the best model. After evaluation, the lowest perplexity, 1.256, was found in the model where n=7. An example of generated method tokens and their probability is shown below:

[('String', 1.0), ('key', 1.0), ('=', 1.0), ('"', 0.9911), ('testKey', 0.1554), ('"', 1.0), (';', 1.0)]

Model Testing: The best-performing model (n=7) was run on the testing set of Java methods, although only the first 100 methods will be explicitly reported. After testing, the average perplexity was calculated to be 1.250.

Training, Evaluation, and Testing on Instructor-Provided Corpus: The same training, evaluation, and testing process was performed on the instructor-provided corpus. After evaluation, the lowest perplexity, 1.763, was found when n=3. Testing on the previous 100 methods, the average perplexity was 1.761.

Output: The outputs for this assignment are stored as .txt files. Each method begins with an ID from 0-99, followed by a set of tuples containing the predicted tokens and their probabilities. A blank line separates each method.

Limitations: The current implementation of the N-gram model makes smoothing/interpolation difficult to include, and thus the average perplexity values are calculated based on successfully generated methods (those with at least one known prediction). This leads to significantly lower perplexity values than anticipated, although the relative trends should remain somewhat accurate. Additionally, the overall sparsity of the models' vocabularies hindered generalization.