CSCI 680 AI for SWE: ASS 1 Report

Jingxiang Sun

jsun20@wm.edu

1. introduction

This project implements an N-Gram language model to predict the next token in a sequence of Java code snippets. The process involves data cleaning, tokenization using Hugging Face's BART tokenizer, and training the model using N-grams with various values of *n*. The project not only focuses on model training but also includes model evaluation using metrics such as accuracy and perplexity. Additionally, multiple N-gram models (1-gram, 2-gram, 3-gram) are compared to identify the best-performing model. The results are validated through accuracy and perplexity calculations, providing a comprehensive analysis of the model's predictive capabilities.

2. Repository Selection

I used SEART GHS tool to find a set of Java repositories based on criteria to include those with at least 4K stars, 5000 commits and 2000 issues. From this set, I got more than 50 repositories and I found a interesting project that is about a pixel game called shattered-pixel-dungeon. This repository got 4.7k stars and definitely is a good repository.

3. Methodology

3.1 Dataset Creation Process

In this assignment, I focused on extracting and processing methods from Java files, which served as the basis for my dataset. The Java methods were collected using the `collect_all_methods()` function. This function walks through the directory of Java files, specifically focusing on the folder "actors", and extracts each method for further processing.

3.1.1 removing comments

As part of data preprocessing, I first implemented a remove_comments() function. This function removes various types of comments from the Java files, including single-line (//), multi-line (/* */), and unnecessary import and package statements. This ensures that only the relevant code content remains for analysis.

3.1.2 Tokenization:

After extracting the cleaned methods, I tokenized the data using Hugging Face's BART tokenizer. Tokenization converts the textual method content into a sequence of tokens that the model can understand. This step was performed in two stages:

① Initial Tokenization: The methods were first tokenized using regular expressions to identify words, symbols, and other relevant tokens. This was replaced by the BART tokenizer, which is more sophisticated and aligned with modern NLP practices.

3.2 Model Training Methodology

To model the tokenized data, I used an N-gram model that was trained on the tokenized methods. The goal was to predict the next token given a sequence of previous tokens (N-grams).

3.2.1 N-Gram Model:

An N-gram is a contiguous sequence of n items from a given dataset. In this project, I implemented 1-gram, 2-gram, and 3-gram models to compare their performance. The NGramModel class was developed to handle these operations. The model stored the frequency of N-grams and calculated the probabilities of the next token based on the previous ones.

3.2.2 laplace smoothing:

To account for unseen N-grams in the test data, I implemented Laplace smoothing. This technique adjusts the probability of unseen N-grams by adding 1 to their counts, ensuring no probability is ever zero.

3.3 Model Evaluation

To evaluate the performance of the N-gram models, two metrics were used: **accuracy** and **perplexity**.

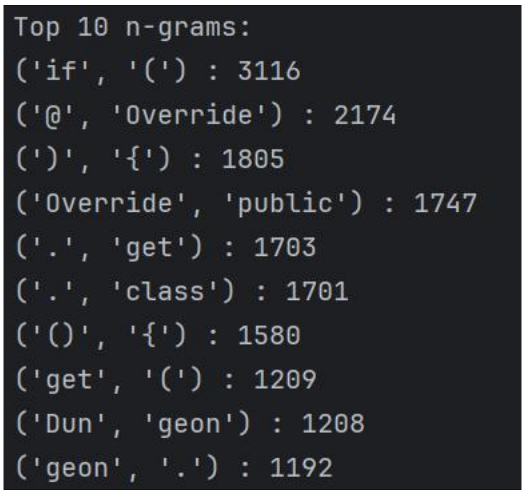
3.3.1 Accuracy

The accuracy metric calculates the proportion of correct n-grams from the test data that the N-Gram model can predict based on its training. Correct predictions are those where the model has encountered the n-gram during training, and accuracy is the ratio of correct predictions to the total number of n-grams.

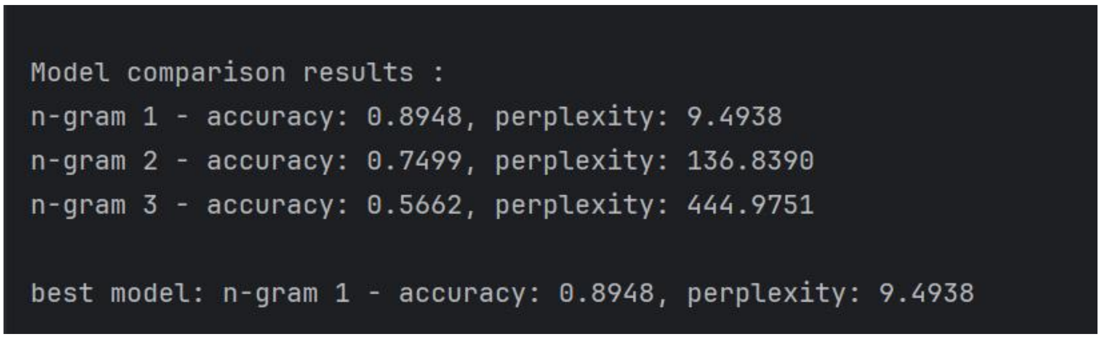
3.3.2 Perplexity

Perplexity measures how well a probabilistic model predicts a sample. It can be understood as the model's uncertainty. A lower perplexity score indicates better performance. Perplexity was calculated using the smoothed probability of the N-grams.

4. Results



```
Top 10 n-grams:
('if', '(') : 3116
('@', 'Override') : 2174
(')', '{') : 1805
('Override', 'public') : 1747
('.', 'get') : 1703
('.', 'class') : 1701
('()', '{') : 1580
('get', '(') : 1209
('Dun', 'geon') : 1208
('geon', '.') : 1192
```



```
Model comparison results :
n-gram 1 - accuracy: 0.8948, perplexity: 9.4938
n-gram 2 - accuracy: 0.7499, perplexity: 136.8390
n-gram 3 - accuracy: 0.5662, perplexity: 444.9751

best model: n-gram 1 - accuracy: 0.8948, perplexity: 9.4938
```