



WILLIAM & MARY

CHARTERED 1693

AI for S/W Eng
Spring 2025

Prof. Antonio Mastropaolo
Assistant: Aya Garryyeva


Recommending code tokens via N-gram models

Your goal is to implement a probabilistic language model (*i.e.*, N-gram), that can assist with code completion for Java systems. In formula:

$$P(t_n | t_{n-1}, t_{n-2}, \dots, t_{n-(N-1)}) = \frac{\text{Count}(t_{n-(N-1)}, \dots, t_{n-1}, t_n)}{\text{Count}(t_{n-(N-1)}, \dots, t_{n-1})}$$

Where t_n and t_{n-1} are sequences of code tokens.

Specifics You are asked to implement an N-gram model to learn the probabilities of tokens sequences based on their occurrences in the training data. This, involves taking decision on various points:

- **Corpus Construction:** The language is fixed *i.e.*, Java (method-level).
- **Context Window:** Consider this as an hyper-parameter of the study that you want to tuned *i.e.*, calibrate.
- **Corpus Preprocessing:** We have seen how important is preprocessing to clean up your data and make it suitable for training.
- **Evaluation of Models:** Intrinsic.
- **Deliverable:** Each team is required to submit a script that accepts a corpus of Java methods as input and outputs for that specific training corpus, the best-performing model, validated on a hold-out portion of the collected elements (*i.e.*, methods not used for the training nor the testing).
- **Examples:** Your script reads from the command line txt files. On each line you have a tokenized method  (which tokenizer to use?). You first extract tokens, then, you project them on a single line. Review the following example:

The core mechanism of your script will start computing probabilities on various configuration of n , for instance $n=3$ and $n=5$.

Using the best-performing model (based on the intrinsic evaluation result), sample sequences of code completion with their respective probabilities for 100 inputs from the test set. What does it mean?

Let's assume that the best-performing model is when $n=7$. Then, we for the first 100 examples of the test set, you use the first 7 code tokens of the model and start making predictions.

How?

Let us assume we are using a 7-gram model to generate tokens by predicting the most likely next token given the previous $n - 1$ tokens. Let C denote a method from the test set, and let C_1, C_2, \dots, C_n represent the code tokens of C . We begin by examining the first six tokens, C_1, \dots, C_6 , and computing $P(C_7 | C_6, \dots, C_1)$. This yields the predicted token PR_7 . Subsequently, we continue generating tokens by

chaining predictions. The process iteratively joins predictions to form multiple contiguous tokens. The generation stops under certain conditions. For instance, we may check if the parentheses of the generated code are balanced. Alternatively, if the computed probabilities stored in a matrix indicate that no valid prediction exists for a given sequence— $P(PR_n | PR_{n-1}, \dots, PR_{n-(n-1)}) = 0$ —we stop at that point.

❓ How many instances do we need to train an n -gram model? It depends whether classes or methods are used. As rule of thumb, ~25K Java methods and ~500 Java classes are considered enough given the task at hand. Reserve a portion of the collected data for the test and eval set. In addition, if you want to push the boundaries to higher values of n (e.g., $n > 10$), the number of elements of your training set has to increase too.

Submission Instructions

📅 **Deadline:** Tuesday March 4th, 2025 @ 12:00 PM.

Each group must submit on Blackboard and GitHub:

1. Submit a 1-page document outlining the dataset creation process, the model training methodology, and evaluation results, including links to the relevant code.
2. Include a README file with instructions on how to configure and execute the script, using a sample Corpus.