# ECS 260: Assignment #3

Due on Tuesday, December 3, 2019, 8pm

*Devanbu FQ2019*

November 15, 2019

## Submission Guidelines and Notes

When submitting your homework, please include the outputs of the commands, plots, and your comments in a pdf. Please include any code you use to get the result, including those for the plots (my suggestion would be to use jupyter notebook, if you are using python to process your result).
Please include comments in your code that explain your decisions and your code structure. Points will be taken off if your code is uncommented, poorly structured, and hard to understand.

## Problem 1

(25 Points) Your first task is to get the language modeling project located here and make sure that you can run the scripts:

`https://github.com/SLP-team/SLP-Core`

Clone the project and carefully read through the README to familiarize yourself with how to set up and run the code. The README includes instructions of how to use the package as a library or as command-line tool.
You will be using the SLP-Core language model, which is a nested language model. It estimates the probability of the next token conditioned on previous tokens, estimating frequencies from a global set of counts augmented with local counts from a *cache* which counts frequencies the current file. It allows additional levels of counts, but you would be using it a two-level mode.

```
java -jar slp-core.jar train-test --train train-path --test test-path -m jm -o 6 --cache
```

where "train-path" is directories named "english_train", "java_train" and "python_train" and the "test-path" is "english_sample" and "java_sample" *etc* in the tarball that we gave you. If you run the command above, you should match the outputs in the case of Java as given in "java-sample-output.txt"

Summarize the last line of the output as

```
Testing complete, modeled * files with * tokens yielding average entropy:  *
```

for each train/sample pair in the provided dataset for english, java, and python.

If you are unable to obtain the same output for java, please contact the TA (wjan@ucdavis.edu) and/or attend office hour.

## Problem 2

(75 points) Now that you have the package working, you will modify some of the commands slightly to explore the effects of different language models.

- (25 points) Using the SLP-Core package, create a script that creates and tests language models at the 1, 2, 3, 4, 5 and 6-gram level *with* the cache enabled. Using 10-fold cross-validation, each case selecting from the data roughly 10% of the projects at random as test, and the remaining as training. Ignore the sample folder from Problem 1 for this part. For each of the 10 test sets in the cross-fold validation, create a box-plot based on the 10 average test set entropies over all the projects in each test set. Put these 6 boxplots (for each n-gram length) together on one plot and comment on how the average test set entropy changes as the n-gram context size is increased. Do this for English, Java, and Python.
- (30 points) Using a 3,4,5-gram models, measure the Mean Reciprocal Rank of the *correct* prediction in Java and Python, for both the *closed category* (keywords, operators, punctuations) and *open category* (identifiers, type names, methods, classes etc). Do this for about 100 randomly chosen files as your

Problem 2 continued on next page. . .

test set; you should have about $O(10^4)$ samples of the reciprocal. Draw 3 boxplots, where each sample is a file, for the closed and open categories, with the n-gram length on the x-asis. You should create 4 plots: *python-closed, java-closed, python-open, java-open* with 3 boxplots in each one.

You can find the precise tokens in the closed category in the files closed_category.txt. Everything else is open.

Hint: Look into the `--verbose` option in the SLP-Core Command-line interface; also you might consider using the `BasicJavaRunner.java` class.

- (20 points) Write a simple pattern matcher to identify method calls *e.g.,* an identifier, followed immediately by an open paren "(". You can assume the way you match for Java is also good enough for Python. Report the MRR in the same settings as the above, but separated by method-calls. These are the tokens most often requested for auto-complete.