COSC 101 C Homework 8: Spring 2025

Introduction

The goal of this assignment is to build a model that can take as input some text and determine whether that text was written by William Shakespeare or Taylor Swift. Concretely, the model will use the frequency of words in the text to determine which artist wrote the text.

This assignment is designed to give you practice with the following topics:

- Program Design
- File I/O
- Dictionaries
- Nested Structures

In addition, you will also get practice with: - Developing modular application - Using `doctests` for test-driven development and to cover many edge cases and - Leveraging from bottom-up implementation and top-down design.

Your assignment

Your task is to complete the following steps:

1. Download `hw8.zip` file and open it. You will see:
    - One python file `hw8_classify.py`
    - One folder `train` with texts for training the model. This folder has sub-folders for `swift` and `shakespeare`.
    - One folder `test` with texts for evaluating the trained model. This folder has sub-folders for `swift` and `shakespeare`.
    - `predictions_toy_gold.txt` which shows what your output file should look like when you run it on some toy data.
    - `glove_vocab.txt` which is the file with all the words in the vocabulary.
2. Complete all of the functions in `hw8_classify.py`. Make sure to build on your bottom-up programming skills and test the functions as you go!
3. Explore the `predictions.txt` and answer specific questions (see Task C)
4. Submit your completed program, the resulting prediction files, and the pdf file with your results. **Please do not change the name of the python file**

Background: Text classification

Text classification is a common task in Natural Langauge Processing (NLP) that is used to assign a class label to some given text. Concretely, in this homework, you will train a model that will assign texts to one of two classes: Shakespeare or Swift.

In order to classify texts, we need to compute some score for the model given each of the different classes, and then pick the class that has a higher score. Here is one approach we can use to calculate the score for a text given some class:

1. Estimate the probability of each word in your vocabulary given the class using some training data. To calculate the probability of some word given a class, count the number of times the word occurs in the training data and divide it by the total number of words. In other words, use the following formula:

P(word | class) = count(word, class train)/count(class train)

2. Given some text you want to classify, compute the score by adding together the *log* probabilities of each of the words in the text given the class (as computed in step 1)

Example output

If your code is running correctly, then when you train your model on the training dataset, and evaluate it on the `swift_toy` and `shakespeare_toy` datasets in the `test` folder, then your code should generate a file that looks like `predictions_toy_gold.txt`.

Apart from the final output, the starter code also has doctests that specify expected behavior for each of the intermediate steps.

Your tasks

Task A: Implement the helper functions

The starter file comes with `doctests` for each function that explicitly specify input-output mappings for the function. In order to make it easier to implement the functions one at a time, all of the function definitions except the first function are commented out. Implement the first function, ensure that it passes all the doctests, before commenting out the next function.

Each helper function should be very short. Feel free to add your own fruitful functions, each of which abstracting a process that you find repeated across the helper function you need to implement.

Note: In writing some of the helper functions, you might have to compose previously written (and tested!) functions

Implement the helper functions in the following order:

- `clean_words`: Takes in a list of words and returns a new list with words in lowercase with the puncutation and other symbols removed.
- `update_frequencies`: Updates the count dictionary (the one used to compte probabilities) with words in some file
- `get_probabilities`: Converts frequencies into probabilities
- `get_logprob_text`: Gets the summed log probability of all the words in a text given a probability dictionary
- `classify`: Given a text returns the class for the text that has the highest log probability.
- `train`: For each class, trains a model for each class (i.e., creates a probability dictionary) given a list of files.
- `classify_texts`: Given trained models, and a list of files, classifies each of the lines in the files and writes all the predictions to an output file.

Task B: Compose the helper functions

Once you have implemented, tested and debugged each these 7 functions, write the `main` function that implements the required functionality. Your main file should create two files: `predictions_toy.txt` which has the predictions for the toy test files, and `predictions.txt` which has the predictions for all the test files. **Note: Some parts of the main function are already created for you**

Task C: Explore the predictions

Open up the `predictions.txt` file in a spreadsheet (e.g., using excel or google sheets). Write up answers to the following questions in a document and convert the document to a pdf file.

1. What is the average accuracy of the model? (you can answer this by taking the mean of the column accuracy in the spreadsheet)
2. Bonus (not required): If you consider the artists separately, what is the average accuracy for Shakespeare and the average accuracy for Swift?
3. Find some examples where the model made incorrect predictions. Speculate why the example might have been misclassified.

Submission

You must submit:

- `hw8_classify.py` with the header completed
- `predictions.txt` and `predictions_toy.txt`
- Completed `reflection.pdf` in which you answer the questions for Task C.