# COSC 101 Homework 9; Spring 2021

The due date for this homework is **Thursday, April 29, 5:00pm EDT**. (Extended from original deadline of Thursday, April 22, 5:00pm EDT.)

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

- Program design
- File I/O
- Dictionaries

## Instructions

Download `hw9.zip` and unzip the compressed file to reveal the following files included with this assignment:

- `hw9.pdf`: this description
- `hw9_small_train.txt`: a text file with just FIXME written comments and numeric ratings; used for debugging your code for training a sentiment model
- `hw9_small_analyze.txt`: a text file with just FIXME written comments; used for debugging your code for applying a sentiment model
- `hw9_amazon_labeled.txt`: a text file with written comments and numeric ratings for products from Amazon; used for training a sentiment model
- `hw9_amazon_analyze.txt`: a text file with written comments for products from Amazon; used for applying a sentiment model
- `hw9_yelp_train.txt`: a text file with written comments and numeric ratings for restaurants from Yelp; used for training a sentiment model
- `hw9_yelp_analyze.txt`: a text file with written comments for restaurants from Yelp; used for applying a sentiment model

All of your work for this assignment will be completed in the file `hw9_sentiment.py`. As with other assignments, this file has a special header at the top with form fields that you should fill out before submitting the code for this assignment. Also, do not change the file name as we sometimes use programs to test your code and they will not be able to find your program and give you any credit if your file is named incorrectly.

Please note that the instructors are aware that there are "solutions" for some aspects of this homework on internet sites such as chegg.com. Remember that copying code from those "solutions" is against syllabus policies and constitutes an academic honor code violation.

### Background: Sentiment Analysis

Sentiment analysis is a form of natural language processing (NLP) that is used to determine the opinion conveyed in a piece of text. For example, sentiment analysis can be used to determine whether a written product review is positive, negative, or neutral.

Conducting sentiment analysis on a review requires identifying features that make a review a positive or negative. For example, a review containing the word "good" is likely positive, while a review containing the word "bad" is likely negative. The context of the word is also often important. For example, "this is not as good as product X" contains the word "good", but "good" is actually referring to a different product and the sentiment for the product being reviewed is negative.

Manually writing a set of rules to determine the sentiment of a review is a difficult task. As an alternative, we can make our sentiment analyzer *learn* the sentiment of a word (positive, negative, or neutral) by analyzing existing reviews. In particular, we can take a set of reviews that include both written comments and numeric ratings (e.g., 1 to 5 stars)–we call this our *training set*–and identify which words are included more often in 5-star reviews, and hence likely convey a positive sentiment, and which words are included more often in 1-star reviews, and hence likely convey a negative sentiment.

More precisely, we can compute a sentiment score for each word in a training set based on the average score of all reviews in which the word occurs. We simply sum the numeric ratings of the reviews in which the word occurs and divide by the total number of occurrences. (Note: if the same word appears more than once in a single review, then we count the word multiple times.) We call the computed word scores our *sentiment model*.

Given a sentimental model, we can compute a numeric rating for a written review without a reviewer-assigned numeric rating. In particular, we compute the average score (from the sentiment model) of all words in the written review. A written review with a high average word score is likely positive, whereas a review with a low average word score is likely negative.

### Required functionality

Your sentiment analysis program must perform two tasks: 1. Train (i.e., create) a sentiment model using a file of written comments and numeric ratings 2. Apply the sentiment model to a file of written comments to compute a numeric rating for each review

### Task 1: Train

The program **must** contain a function called `train` that takes the name of a file containing training data (i.e., written comments and numeric ratings) and returns a sentiment model (i.e., a dictionary whose keys are words appearing in the training reviews and whose values are the sentiment scores for each word). Each word's sentiment score should be computed using the methodology described above.

Assume the file of training data contains one review per line. Each line contains a numeric rating, followed by a space, followed by the written comment. See `hw9_small_train.txt` for an example. If the training file cannot be read (i.e., an exception occurs), then the `train` function **must** print the error message `Unable to read the training file` and return an empty sentiment model (i.e., an empty dictionary).

You can use the provided `get_words` function to get a list of words (in lowercase without punctuation) from a string.

Your program **must** contain at least one additional *helper function* that is called by the `train` function.

## Task 2: Analyze

The program **must** contain a function called `analyze` that takes the sentiment model (i.e., the dictionary of words and sentiment scores) and the name of a file containing written comments (without numeric ratings). The function should return a list of tuples, where each tuple contains a numeric rating–computed using the methodology described above–and the (original) written comment. Ignore words in a written comment that do not appear in the model.

Assume the file of written comments contains one review per line. See `hw9_small_analyze.txt` for an example. If the file cannot be read (i.e., an exception occurs), then the `analyze` function **must** print the error message `Unable to read analyze file` and return an empty list.

Your program **must** contain at least one additional *helper function* that is called by the `analyze` function.

### Main

The program **must** contain a `main` function that asks for the name of the file of training data and the name of the file of written comments to which the sentiment model should be applied.

The program **must** output: 1. The word and sentiment score for every word in the sentiment model. The words **must** be displayed in alphabetical order. 2. The numeric rating and (original) written comment for every review to which the sentiment was applied. The reviews **must** be displayed in the order they appear in the file of written comments, and the numeric rating **must** be rounded to a single decimal point.

Your program **must** contain at least one additional *helper function* (besides `train` and `analyze`) that is called by the `main` function.

Here is an example of how the output from your `main` function should look. (`hw9_small_train.txt` and `hw9_small_analyze.txt` are inputs entered by the user.).

```
Enter training filename: hw9_small_train.txt
a 3.0
advertised 1.0
am 2.0
as 1.0
broke 2.0
dissatisfied 2.0
does 1.0
function 1.0
functions 4.0
good 5.0
i 2.0
is 3.5
junk 1.0
made 5.0
mediocre 3.0
not 1.0
product 3.0
this 3.0
well 4.5
Enter analyze filename: hw9_small_analyze.txt
3.5 This product works very well.
2.7 This product is a piece of junk.
4.0 Very poorly made product!
```

### Challenge Problems

*Stop words* are a set of commonly used words: e.g., 'the', 'is', 'a'. These words are typically ignored when performing sentiment analysis, because they occur frequently in both positive and negative reviews. Modify the `train` and `apply` functions to take a list of stop words, and ignore these stop words in written comments. Also modify the main function to ask for the name of a file containing stop words, with one stop word per line. (You will need to create or download a file of stop words to test your code.)

## Testing

There is no "separate" testing program for this homework, but there are doctests written for the `get_words`, `train`, and `analyze` functions in the `hw9_sentiment.py` file.

You may choose to add to the existing doctests, but you are not required to do so. Note that the `main` function is set up to invoke the doctests.

## Submission Instructions

Please upload your `hw9_sentiment.py` file.

Remember to complete the questions at the top of the `hw9_sentiment.py` file and that the file you submit needs to have this exact filename.

## Grading

Your assignment will be graded on two criteria:

1. Correctness: [75%]. The correctness part of your grade is broken down as follows:

| Category (function) | Portion of grade |
|---|---|
| Correctly trains a sentiment model | 30% |
| Correctly applies a sentiment model | 30% |
| Correctly displays the sentiment model and the reviews to which it was applied | 15% |

2. Program design and style [25%]. For this assignment, you are tasked with completing three functions and writing **at least** three additional helper functions. Within each function:
    - Variable names should be meaningful
    - Code should contain at least a few descriptive comments if it is complex. Do *not* comment every line of code with low level explanations of what each line does. Focus on high level ideas. You will **lose points** if you document every line of the file.
    - Functions should contain meaningful docstrings.