

COSC 101 Homework 8: Spring 2024

The due date for this homework is **Monday, April 29th, 11:00pm**.

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

- Program design
- File I/O
- Dictionaries

Instructions

Download [hw8.zip](#) and unzip the compressed file to reveal the following files included with this assignment:

- `hw8.pdf`: this description
- `hw8_test_train_*.txt`: multiple files to be used as inputs when testing `hw8_train.py`
- `hw8_test_analyze_*.txt`: multiple files to be used as inputs when testing `hw8_analyze.py`
- `hw8_small_train.txt`: a text file with just written comments and numeric ratings; used for input when testing your `hw8_train.py`
- `hw8_small_model.txt`: a sentiment model for `hw8_small_train.txt`; used as output for debugging your `hw8_train.py` and as input when testing `hw8_analyze.py`
- `hw8_small_analyze.txt`: a text file with just written comments; used for testing your `hw8_analyze.py`
- a folder called `larger_test_files` with the following content:
 - `hw8_yelp_train.txt`: a text file with written comments and numeric ratings for restaurants from Yelp; used for training a sentiment model
 - `hw8_yelp_analyze.txt`: a text file with written comments for restaurants from Yelp; used for applying a sentiment model
 - `hw8_yelp_model.txt`: a sentiment model for yelp reviews; used for debugging your `hw8_analyze.py`
 - `hw8_amazon_labeled.txt`: a text file with written comments and numeric ratings for products from Amazon; used for training a sentiment model
 - `hw8_amazon_analyze.txt`: a text file with written comments for products from Amazon; used for applying a sentiment model

All of your work for this assignment will be completed in two files: `hw8_train.py` and `hw8_analyze.py`. As with other assignments, these files have a special header at the top with form fields that you should fill out before submitting the code for this assignment. `hw8_train.py` is used in **Part 1**, and `hw8_analyze.py` is used in **Part 2**.

Background: Sentiment Analysis

Sentiment analysis is a task common to 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.

Part 1

Training a Sentiment Model

Your task is to design a program (in `hw8_train.py`) that trains a sentiment model on existing reviews. Your 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 `hw8_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).

Additionally your program should include (and use) a function which saves your sentiment model in a txt file. For example, training a model on `hw8_small_train.txt` should result in a file with the following content (i.e. the trained sentiment model):

```
this 3.0
product 3.0
is 3.5
good 5.0
well 4.5
made 5.0
functions 4.0
a 3.0
mediocre 3.0
broke 2.0
i 2.0
am 2.0
dissatisfied 2.0
junk 1.0
does 1.0
not 1.0
```

```
function 1.0
as 1.0
advertised 1.0
```

Notice the formatting: A saved model should have each word and its sentiment score on a separate line (with a space between the word and score).

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.

Part 2

Analyzing Reviews with a Sentiment Model

Your task is to design a program (in `hw8_analyze.py`) which uses a saved sentiment model to assign ratings to reviews. The program **must** contain a function called `analyze` that takes the filename of a saved 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 lists, where each sublist 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 `hw8_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.

A couple of saved sentiment model are included with the homework (`hw8_small_model.txt` and `hw8_yelp_model.txt`). Your `hw8_analyze.py` should save the reviews and their predicted ratings to a file.

For example, using `hw8_small_model.txt` and determining the ratings for the `hw8_small_analyze.txt` would yield:

```
3.5 This product works very well.
2.7 This product is a piece of junk.
4.0 Very poorly made product!
```

And, using `hw8_yelp_model.txt` and determining the ratings for the `hw8_small_analyze.txt` would yield:

```
3.679178895602135 This product works very well.
3.570560307210949 This product is a piece of junk.
3.142216530929244 Very poorly made product!
```

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

Challenge Problem (OPTIONAL)

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 your programs to take a list of stop words, and ignore these words in written comments. Your code should take 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).

Do Not modify `hw8_train.py` and `hw8_analyze.py`. Instead copy the initial code in new files called `hw8_train_challenge.py` and `hw8_analyze_challenge.py`

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 saves 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 functions described in the assignment and writing 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 with test cases