CSE 5243: Homework #3

Deadline: **11:59PM on 10/24/2022**.
No late submissions will be accepted.

Instructions.

You will have around 3 weeks to work on this programming assignment. We currently use a 110-point (100 regular + 10 bonus point) scale for this homework, but it will take 10% of your final grade.

What you should turn in:

Please turn your code and all documentations/reports to Carmen. Please try to be considerate of the TA and make sure he can run your code with minimal effort. :-)

Questions?

Please create a post on Carmen discussion areas or Microsoft Teams group to get timely help from other students, the TA, and the instructor. Remember that participation takes 5% of your final grade. You can show your participation by actively answering others' questions! Besides, everyone can benefit from checking what have been asked previously. Please try to avoid directly sending emails to the instructor, as she receives a lot of emails everyday and no timely reply is guaranteed. Thanks!

1 Task: Sentiment Classification (100 points)

This assignment is the second part of a longer-term project. The objective is to give you the experience of building classifiers for your preprocessed real data in HW2.

1.1 Review HW2.

What you have done for HW2:

- **Data**: Sentiment Labelled Sentences Data Set, which contains sentences labelled with positive or negative sentiment. It can be downloaded here `http://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences`.

Read their readme.txt file for detailed information. There are three subsets respectively from IMDB, Amazon and Yelp. Please merge them as a single dataset, which should contain 3,000 sentences in total.

- **Data Format**: Each data file is .txt where each row has two columns: sentence body and sentence label. For example, one sample sentence is "Very little music or anything to speak of. 0", where the first column "Very little music or anything to speak of." is the content of a sentence while the second column "0" is its sentiment label (1 means positive; 0 means negative).

- **Results from HW2**: In HW2, you should have constructed a feature vector for each sentence in the data set. Let us assume you use the frequency of unique words in the sentence body to construct a feature vector. If there are totally M sentences and N unique words in the dataset, you should have constructed a $M \times N$ matrix D, where $D_{i,j}$ means the count of word j in sentence i.

Please note that your feature vector for each sentence should only be based on the sentence content, and does not contain the sentiment label.

1.2 Task Description for HW3.

1.2.1 Overall goal.

The goal of HW3 is to build and evaluate classifiers to predict the sentiment of a given sentence. The true label of each sentence is binary (i.e., positive vs negative) and given in the dataset.

1.2.2 Dataset split.

To build a classifier, you need a training, validation, and testing set. Now that you have 3,000 sentences in total, you can randomly sample p% of them as training, q% as validation, r% as testing, where p+q+r=100. For example, randomly sample 70% as training, 15% as validation, 15% as testing, or other percentage you prefer. Please detail how you split your data in your report.

1.2.3 Feature selection.

You will also play around with methods to select "important" features. Let us assume you start with a feature vector that comprises or keeps track of 2048 words. What you will need to do is to identify and leverage a measure of importance to pare down the feature vector size down to, say, 256 or 512 words, and compare and contrast the performance of **the original feature set vs pruned feature set** on two classification algorithms. **Please note that these numbers are completely made up and feel free to decide how many features you want to use originally and after pruning.**

You have the freedom to define your own measure of importance. You may consider one of the following two intuitive choices:

(1) One intuitive measure is the overall frequency of a word in the entire dataset. You can keep the top-K most frequent words (e.g., top 1000 most frequent words in the entire dataset) and prune less frequent ones. Or, you can keep words that appear at least K times (e.g., K=2) in the dataset. If you choose to use this kind of measure, you may first remove stop words like "the" and "a" from your dataset, as they are not informative but very frequent.

(2) TF-IDF. IF-IDF is the product of two statistics, term frequency and inverse document frequency. The tf-idf value increases proportionally to the number of times a word appears in the document, but is often offset by the frequency of the word in the entire corpus, which helps to adjust for the fact that some words appear more frequently in general. For more information, please refer to `https://en.wikipedia.org/wiki/Tf%E2%80%93idf`.

For word t that occurs in sentence d, you can calculate its tf-idf in the simplest way in the previous link:

- $\text{tf}(t, d) = f_{t,d}$, where $f_{t,d}$ is simply the frequency of word t in document d.
- $\text{idf}(t, D) = \log(\frac{N}{|d \in D : t \in d|})$, is a measure of whether word t is common or rare across all sentences D. Note in your experiment, you can use your training sentence set as D. It can be obtained by dividing the total number of sentences (i.e., N) by the number of sentences containing the term (i.e., $|d \in D : t \in d|$), and then taking the logarithm of that quotient.
- Finally, the tf-idf score of word t in sentence d, is $\text{tf}(t, d)$ * $\text{idf}(t, D)$.

Note that you have the freedom to decide what importance measures to use, how many features should be kept, etc. Please detail what you used in your report.

1.2.4 What classification algorithms to use?

You are expected to test out **two** different classifiers (K-nearest neighbors, decision trees and Naïve Bayes are your possible choices). You should implement at least one of the classifiers on your own and use free software/packages for the other (if any).

Our ***strong recommendation*** is that after you implement the classifier(s) by yourself, you also try existing software/packages for the same algorithm(s) to make sure you have correct implementation(s), unless you are very confident. This is not required, but highly encouraged. Keep in mind that if you use existing software, you may have to transform your feature vector dataset to match their input specifications. Your report should describe any further data transformations you have had (if any) in order to work with these software packages.

Bonus task (10 points): In case you have got much experience, you are encouraged to test a third classifier using more advanced algorithms (such as

SVM or a feedforward neural network or an LSTM network[1] for textual sequence data) with the help of popular Python libraries such as Tensorflow (`https://www.tensorflow.org/`) and Pytorch (`https://pytorch.org/`).

Note: No matter you choose to do the bonus task or not, make sure you implement at lease one classifier on your own by following the algorithms/pseudo code. For example, simply calling a built-in function of some Python package is NOT your own implementation. Similarly, if you want to test the LSTM network, using existing LSTM cells is not regarded as your implementation, but you can still earn the bonus points, if you also evaluate another two basic classifiers and one of them is implemented by yourself.

1.2.5 What results to expect?

To sum up, assuming you pick two classifiers (CL-1; CL-2) and you have the original feature vector from the last assignment (FV-1) (see notes below on things you may need to correct) and a pared down feature vector where you have selected some features (FV-2) as discussed in Section 1.2.3. You will need to run 4 sets of experiments (**each classifier on each type of feature vector** – think of it as a 2X2 experiment configuration matrix).

For each set of experiments (configuration) you will need to report:

(1) the overall scalability of the resulting approach (time to build classifier model – this is also known as the offline efficiency cost)

(2) time to classify a new tuple (also known as the online efficiency cost)

(3) the accuracy of the classifier respectively on training, validation and testing sets.

(4) **comments on the advantages (or disadvantages, depending on what you find) using the features and the method you adopted**. Examples could include – feature selection helps improve accuracy and efficiency costs, or feature selection is ineffective. Back up your observations with numbers/graphs/charts.

Notes: (1) Please note that if the TA provides feedback on your preprocessing step and obtained features in HW2, please try to address the feedback in this assignment. He should be able to finish grading Problem 5 in HW2 by Oct 12th, but please don't wait up but start building classifiers and the whole evaluation pipeline now; you could simply update your feature set later. (2) If you have a third advanced classifier, to get the 10 bonus points, it is fine to just test it under only one feature setting, i.e., either FV-1 or FV-2, but you need to report (1) -(3) and comment on (4) as well.

[1] `https://adventuresinmachinelearning.com/recurrent-neural-networks-lstm-tutorial-tensorflow/`

1.3 What You Should Turn in.

Use the submit command to submit all source files (such as README, source code, report, etc.), as your did for HW2:

- All source code and data files. Make sure the data files and the program are in the same folder. Please make it easy to run your code (e.g., have a test run by yourself).

  You are also encouraged to add more comments in your code. In case your code cannot run or the results are wrong, the TA can still give you partial credit if the overall procedure is correct.

- A detailed README file that contains all the information about the directory, e.g., how to run your program, how to interpret the resulting output, etc. **You can decide what your code outputs in the terminal to show the TA.** Examples would be (1) listing a few sentences, their predicted labels and their true labels; (2) a summary of your classifiers' results, such as their accuracy for each set of experiments.

- **A detailed report, listing all underlying assumptions (if any), and explanations for performance obtained is expected. Prior to your submitting this report, if the TA provides feedback on the previous programming assignment in HW2, please try to address it in this assignment's report**.

Detailing what you did is very important even if it did not work. Please describe any difficulties you may have encountered and any assumptions you are making.

1.4 Additional Notes

Regarding programming languages, Python, C++ or Java highly encouraged. Feel free to directly submit the files through the Carmen website rather than use command lines.

1.5 Acknowledgement

The dataset was originally used in the following paper:
[1] "From Group to Individual Labels using Deep Features", Kotzias et al., SIGKDD 2015.