CSCI544: Homework Assignment №2

Due on February 23, 2026 (11:59 PM)

This assignment is an extension to HW assignment 1 on sentiment analysis. **Please create your own Jupyter Notebook for this HW**: it should contain both code and text cells with sufficient comments such that the reader can understand your solution as well as your responses for some of the questions. Make sure to clearly indicate the question number before the corresponding codes, such as "Question 1", "Question 2(a)", etc.. On the Jupyter notebook, please print the requested values, too.

Please follow the instructions and submit:

1. A PDF report that contains answers to the questions in the assignment, along with brief explanations about your solution. Please also print the completed Jupyter Notebook in PDF format and merge it with your report. (Just submit one PDF file by merging your written answer and the completed Jupyter notebook.) In your completed Jupyter notebook, please also print the requested values.

2. You also need to submit an executable .py file, which, when run, generates the requested numerical outputs in the assignment as listed at the end of the assignment description. We need the .py file to check code overlaps and detect plagiarism. Please include the Python version you use.

The preferred library to implement neural models is PyTorch but TensorFlow (or Keras) is also acceptable. Please name your PDF and .py files as "HW2-YourFirstName-YourLastName-AAA.pdf/py", where "AAA" is either "PyTorch" or "TensorFlow" depending on the library you have used.

For reproducibility, you must use `random state = 42` whenever applicable (e.g., sampling data, train-test split, and model initialization).

1. Dataset Generation (5 points)

We will use the Amazon reviews dataset used in HW1. Load the dataset and build a balanced dataset of 250K reviews along with their ratings (50K instances per each rating score) through random selection. Use the following rules to create ternary labels from the ratings: We assume that ratings more than 3 denote `positive sentiment` (class 1) and rating less than 3 denote `negative sentiment` (class 2). Reviews with rating 3 are considered to have `neutral sentiment` (class 3). You can store your dataset after generation and reuse it to reduce the computational load. For your experiments consider a 80%/20% training/testing split.

2. Word Embedding (30 points)

In this part the of the assignment, you will learn how to generate two sets of Word2Vec features for the dataset you generated. You can use the Gensim library for this purpose.

(a) (10 points)

Load the pretrained "word2vec-google-news-300" Word2Vec model and learn how to extract word embeddings for your dataset. Try to check semantic similarities of the generated vectors using two examples of your own, e.g., $King - Man + Woman = Queen$ or $excellent \sim outstanding$.

(b) (20 points)

Train a Word2Vec model using your own dataset. Set the embedding size to be 300 and the window size to be 11. You can also consider a minimum word count of 10. Check the semantic similarities for the same two examples in part (a). What do you conclude from comparing vectors generated by yourself and the pretrained model? Which of the Word2Vec models seems to encode semantic similarities between words better?

3. Simple models (20 points)

Using the Word2Vec features that you can generate using the two models you prepared in the Word Embedding section, train a perceptron and an SVM model similar to HW1 for class 1 and class 2 (binary models). For this purpose, you can just use the average Word2Vec vectors for each review as the input feature ($x = \frac{1}{N}\sum_{i=1}^{N} W_i$ for a review with N words). To improve your performance, use the data cleaning and preprocessing steps of HW1 to include only important words from each review when you compute the average $x = \frac{1}{N}\sum_{i=1}^{N} W_i$. Report your accuracy values on the testing split for these models for each feature type along with values you reported in your HW1 submission, i.e., for each of perceptron and SVM, you need to report two accuracy values for "word2vec-google-news-300" features and your own Word2Vec features.

What do you conclude from comparing performances for the models trained using the two different feature types?

4. Feedforward Neural Networks (25 points)

Using the features that you can generate using the models you prepared in the Word "Embedding section", train a feedforward multilayer perceptron network for sentiment analysis classification. Consider a network with two hidden layers, each with 50 and 10 nodes, respectively. You can use cross entropy loss and your own choice for other hyperparamters, e.g., nonlinearity, number of epochs, etc. Part of getting good results is to select good values for these hyperparamters.

You can also refer to the following tutorial to familiarize yourself:

`https://www.kaggle.com/mishra1993/pytorch-multi-layer-perceptron-mnist`

Although the above tutorial is for image data but the concept of training an MLP is very similar to what we want to do.

(a) (10 points)

To generate the input features, use the average Word2Vec vectors similar to the "Simple models" section and train the neural network. Train a network for binary classification using class 1 and class 2 and also a ternary model for

the three classes. Report accuracy values on the testing split for your MLP model for each of the binary and ternary classification cases.

(b) (15 points)

To generate the input features, concatenate the first 10 Word2Vec vectors for each review as the input feature ($x = [W_1^T, ..., W_{10}^T]$) and train the neural network. Report the accuracy value on the testing split for your MLP model for each of the binary and ternary classification cases.

What do you conclude by comparing accuracy values you obtain with those obtained in the "'Simple Models" section (note you can compare the accuracy values for binary classification).

5. Convolutional Neural Networks (20 points)

Using the vectors you prepared in the "Word Embedding" section, train a convolutional neural network (CNN) for sentiment analysis classification.

Train a simple CNN for sentiment analysis. You can consider an two-layer CNN with the output channel sizes of 50 and 10. To feed your data into the CNN, limit the maximum review length to 50 by truncating longer reviews and padding shorter reviews with a null value (0). You can use cross entropy loss and your own choice for other hyperparamters, e.g., nonlinearity, number of epochs, etc. Train the CNN network for binary classification using class 1 and class 2 and also a ternary model for the three classes. Report accuracy values on the testing split for your CNN model.

Note that in total, you need to report **16** accuracy values:

- (2 Word2Vec features)*(Perceptron + SVM models)*(binary classification setting) = 2*2*1 = 4 cases
- (2 Word2Vec features)*(2 FFNN + 1 CNN)*(binary + ternary settings) = 2*3*2 = 12 cases