Programming Assignment 2
(Submit via Blackboard by 6th March, 2020)

In this assignment, you will build feed forward neural networks for sentiment classification and language modelling. We will use the same data from Assignment 1.

1. Build your feed forward neural network for sentiment classification.

We are defining sentiment classification as two classes: positive and negative. Our data set consists of airline reviews. The zip directory for the data contains training and test datasets, where each file contains one airline review tweet. You will build the model using training data and evaluate with test data.

a. Create your Vocabulary: Read the complete training data word by word and create the vocabulary V for the corpus. You must not include the test set in this process. Remove any markup tags, e.g., HTML tags, from the data. Lower case capitalized words (i.e., starts with a capital letter) but not all capital words (e.g., USA). Remove all stopwords. You can use appropriate tools in nltk to stem. Stem at white space and also at each punctuation. In other words, "child's" consists of two tokens "child and 's", "home." consists of two tokens "home" and ".". Consider emoticons in this process. You can use an emoticon tokenizer, if you so choose. If yes, specify which one.
b. Extract Features: Convert documents to vectors using tf-idf representation. Do not use existing tf-idf calculation library.
c. Training: Note that if you want to experiment with different stemmers or other aspects of the input features, you must do so on the training set, either through cross-validation or by setting a development set from the get-go. You must not do such preliminary evaluations on test data. Once you have finalized your system, you are ready to evaluate on test data. You can ignore any words that appear in the test set but not the training set.
d. Build and train a feed forward neural network: Build your FFNN with 2 layers with hidden vector size 20. Initialize the weights with random numbers. Use mean squared error as your loss function, sigmoid as the activation function. You can start training with learning rate of 0.0001. If you would like to tune any parameters, you must do so using cross-validation on the training data only. Once you have finalized your system, you are ready to evaluate on test data.
e. Evaluation: Compute the most likely class for each review in the test set using each of the combinations of stemming + tf-idf, no-stemming + tf-idf. Compute and report accuracy with confusion matrix on a .txt or .log file.

2. Build your feed forward neural network for language modelling.

We will treat language modeling as a binary classification of positive and negative n-grams, for n=2. Positive n-grams belong to the language model, whereas negative ones do not. You will create the positive and negative n-grams from the provided data as follows:

a. Pre-processing: Read the data word by word from a "train/positive". Remove any markup tags, e.g., HTML tags, from the data. Lower case capitalized words (i.e., starts with a capital letter) but not all capital words (e.g., USA). Do not remove stopwords. Tokenize at white space and also at each punctuation. Consider emoticons in this process. You can use an emoticon tokenizer, if you so choose. If yes, specify which one.
b. Construct your n-grams: Create positive n-gram samples by collecting all pairs of adjacent tokens. Create 2 negative samples for each positive sample by keeping the first word the same as the positive sample, but randomly sampling the rest of the corpus for the second word. The second word can be any word in the corpus except for the first word itself.
c. Build and train a feed forward neural network: Build your FFNN with 2 layers with hidden vector size 20. Initialize the weights with random numbers. Use mean squared error as your loss function, sigmoid as the activation function. You can start training with learning rate of 0.0001 or 0.00001. If you would like to tune any parameters, you must do so using cross-validation on the training data only. Once you have finalized your system, you are ready to evaluate on test data.
d. Evaluate: Compute the most likely class for each n-gram in the test set. Use the test set in the "test/positive" folders. Save your results in a.txt or .log file.
e. Evaluation Methods: Compute and report accuracy. Save your output and accuracy in a .txt or .log file.

Documentation and Deliverables:

a. Documentation: Use the same documentation format from Assignment 1. Start all your files with a description of your code. Write short description of each function on top of it.
b. Deliverables: Submit a zip file named with student1[firstname initial][lastname]_student2[firstname initial][lastname]_[hw#].zip (I.e. student 1 jamie lee, student 2 kahyun lee: jlee_klee_h2.zip).
c. Zip file should include: Your code(s), and .log file or .txt file that contains your output and another file that contains accuracy. You can choose whatever is convenient for you. Log can be created using logging library. Txt file can be created using simple io library.