Natural Language Processing [CS4120]
Assignment 2

Instructor: Professor Lu Wang

Deadline: February 28th, 2020 at 11:59pm on Blackboard

For the programming questions, you can use Python (preferred), Java, or C/C++. Please include a README file with detailed instructions on how to run your code. Failure to provide a README file will result in **deduction of points** (5 to 10 points per problem). Your final deliverable for this homework should consist of one zipped folder which contains three folders inside (one folder per problem), and each folder must contain the following things:
i) Text files with your answers for questions requiring textual answers (name each file as instructed per question)
ii) README describing how to run your code for each of the programming problem.
iii) Code scripts for programming questions. It is recommended to comment your code at important steps to avoid any ambiguity while grading.

This assignment has to be done individually. If you discuss the solution with others, you should indicate their names in your submission. If you use ideas/code from any online forums, you should cite them in your solutions as well. **Violation of the academic integrity policy is strictly prohibited, and any plausible case will be reported once found. No exceptions will be made.**

1 Parsing [25 points]

In class, we explored parsing algorithms. Now, you will apply an available tool to parse a given corpus. For this problem, you will use Spacy: https://spacy.io/, to parse the following dataset:
http://bit.ly/2RMtGSm
This corpus contains sentences each on a new line. Your task is: Combine the text from all the given sentences into one raw text variable. Next, process this raw text using spacy to obtain the part-of-speech tags, constituent parse trees, and named entities.

Report your answers in file named Q1.txt. Please submit your code along with a README file explaining how to run your code.

1.1 [3 points]

Report the number of sentences parsed; do so by searching for "ROOT" in the parses or use the sentence segmentation method of spacy parser.

1.2 [4 points]

Using the part-of-speech tags that the parser has given you, report the average number of verbs per sentence in the overall corpus (i.e. total number of verbs divided by total number of sentences in the corpus). Use the total number of sentences from 1.1. Also report the part-of-speech tags that you are using to identify the verbs.

1.3 [5 points]

Using the parse trees that the parser has given you, report the total number of prepositions found in the overall corpus. In addition, report top three most commonly used preposition.

1.4 [5 points]

Using the named entity recognition method of the parser, report the total number of entities found in the overall corpus. Also, list out all the unique entity labels found in the corpus (e.g. PER, ORG, etc.).

1.5 [8 points]

Take a look at the parsing results. List out two common errors made in each type of parsing results, and briefly discuss potential methods to reduce each type of error.

2 CKY parser [20 points]

As shown in class, CKY is a parsing algorithm for context-free grammars which employs bottom-up parsing and dynamic programming. Probabilistic CKY allows to recover the most probable parse tree given the probabilities of all productions. Using the rules and probabilities given below, draw a probabilistic CKY chart for the sentence :

"assiduous student studies throughout the night".

Please submit your chart in the format as shown in lecture slides in either pdf format or image format (png or jpg). Hint: You could create your chart using an excel sheet (each cell contains a block) and then export it as a pdf or take a screenshot of it. For example, you can refer to the sample chart as shown here http://bit.ly/2NN0SaP.

The rules and probabilities are as follows:

- $S \rightarrow DP\ NP$ 1.0
- $DP \rightarrow Det\ NP$ 0.3
- $DP \rightarrow Adj\ NP$ 0.3
- $DP \rightarrow N\ N$ 0.2
- $DP \rightarrow PP\ DP$ 0.3
- $DP \rightarrow P\ DP$ 0.2
- $NP \rightarrow NP\ VP$ 0.3
- $NP \rightarrow NP\ P$ 0.2
- $NP \rightarrow NP\ PP$ 0.2
- $VP \rightarrow V\ DP$ 0.3
- $VP \rightarrow VP\ P$ 0.1
- $PP \rightarrow P\ Det$ 1.0
- $P \rightarrow throughout$ 1.0
- $Det \rightarrow the$ 1.0
- $NP \rightarrow assiduous$ 0.2
- $NP \rightarrow student$ 0.3
- $NP \rightarrow night$ 0.2
- $NP \rightarrow studies$ 0.3
- $Adj \rightarrow assiduous$ 1.0
- $V \rightarrow night$ 0.3
- $V \rightarrow studies$ 0.8
- $VP \rightarrow night$ 0.3
- $VP \rightarrow studies$ 0.3

3 Sentiment Analysis [55 points]

Background: Sentiment analysis refers to the use of natural language processing to systematically identify, extract, quantify, and study affective states and subjective information. As shown in class it is widely applied to reviews and survey responses, and social media content for applications that range from marketing to customer service. In this problem, we will explore a basic task in sentiment analysis, which is classifying the polarity of a given text to be positive or negative.

Dataset: For this problem, we will use the training data containing 10,000, movie reviews collected by Cornell University, where each review is labelled based on their ratings. Reviews for training dataset are in the corresponding files ('train_negative_reviews.txt' for negative reviews and 'train_positive_reviews.txt' for positive reviews). Assign the negative sentiment reviews from the negative review file with labels of 0 and the positive sentiment reviews from the positive reviews file as 1, these 0 and 1 labels will be your target labels for sentiment analysis.

The training data is available at: http://bit.ly/2GeYWnr

You can use the following tools for implementation:

1. PyTorch http://pytorch.org/
2. Scikit-learn http://scikit-learn.org/stable/index.html
3. TensorFlow https://www.tensorflow.org/

In addition to output, results, and your textual answers (a separate text file for each sub-question e.g. Q3.1-results.txt, Q3.2-results.txt), please submit your code along with a README file explaining how to run your code.

3.1 [10 points]

Train a Multilayer Perceptron (MLP) to predict sentiment score using unigram features (e.g. word counts or one-hot encoding of words; one-hot encoding means binarize occurrences, 1: occurs in the review, 0: does not occur) as input. You can use the training and testing functions for MLP from the tool of your choice. You are not required to implement the learning algorithm (i.e., back-propagation) for this problem. Your MLP should have an input layer, two hidden layers, and an output layer; the second hidden layer should have 10 nodes. Use 10-fold cross-validation to optimize parameters (e.g. activation function or number of nodes in the first hidden layer). Use accuracy as the metric for parameter selection.

Note: Implement the 10-fold cross validation and unigram features without using any external library.

Your output text file must contain the following:

1. Description of your optimization process. (What parameter are you optimizing?)
2. List out the values of the parameters for optimization.

3. Report accuracies of your model on all chosen parameters.

4. Report the chosen parameter(s) and explain why.

3.2 [5 points]

Use the chosen parameters from 3.1, re-train the model on the **whole training set**, and report the accuracy on the same entire **training** set. (Doing this tells you how the selected model performs on the full training data, and this will likely provide an "upperbound" performance on test data.)

3.3 [10 points]

You will now use word embeddings for each word to create features for each review. Use the popular Google pre-trained word embeddings GoogleNews-vectors-negative300.bin.gz from Word2vec which have been trained on millions of Google News articles.
You can download the pre-trained word vector binary file from:
https://github.com/mmihaltz/word2vec-GoogleNews-vectors
Hint: To compute the feature vector of each review, you can take the average of all word embeddings of words in that review.

Do the same as described in 3.1: Use 10-fold cross-validation to optimize parameters (e.g. activation function or number of nodes in the first hidden layer). Use accuracy as the metric for parameter selection. Report the parameters of your best model. Then re-train the best performing model on the whole training set, and report the accuracy on the same **training** set.

Your output text file must contain the following:

1. Description of your optimization process. (What parameter are you optimizing?)

2. List out the values of the parameters for optimization.

3. Report accuracies of your model on all chosen parameters.

4. Report the chose parameter(s) and explain why.

5. The accuracy of your model on the entire training set using this parameter.

3.4 [20 points]

Next, you will compute the feature vectors for each review using the popular TF-IDF method and do dimensionality reduction using SVD to reduce the number of features.
In short, to compute training set's review feature vector: i) first generate the TF-IDF vectors of the reviews, and ii) then apply Singular Value Decomposition (SVD) on it.

Do the same as described in 3.1: Use 10-fold cross-validation to optimize the SVD's **number of components** to use and model's parameters (e.g. activation function or number of nodes in the

first hidden layer). Use accuracy as the metric for parameter selection. Report the parameters of your best model. Then re-train the best performing model on the whole training set, and report the accuracy on the **training** set.

Hint: You can use the SVD implementation of Sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html. The *n_components* parameter is the **number of components**.

Your output text file must contain the following:

1. Description of your optimization process. (What parameter are you optimizing?)
2. List out the values of the parameters for optimization.
3. Report accuracies of your model on all chosen parameters.
4. Report the chose parameter(s) and explain why.
5. The accuracy of your model on the entire training set using this parameter.

3.5 [10 points]

Using the best model from above (based on results from 3.2, 3.3., and 3.4), predict the sentiment scores for all 2000 reviews in this test set: http://bit.ly/30JWpeo

Classify the reviews into two categories based on your predicted sentiment score. Submit a single text file, which contains the labels for test data movie reviews as predicted by your best model which each prediction on a separate line (each line contains a 0 or 1). (For grading purpose, please DO NOT change the order of test data and the corresponding labels, for example, the first label in your submitted file must be the prediction for the first review in the test dataset.)