Indian Institute of Technology Kanpur
CS671: Natural Language Processing, 18–19

Student Name: Gurpreet Singh
Roll Number: 150259
Date: March 23, 2018

ASSIGNMENT 2

1. Problem Statement

Classifying a set of documents to predict the sentiment of the document.

The task at hand is a binary classification problem on a set of movie reviews provided in the Large Movie Review Dataset. The prediction label is the positivity/negativity of the review.

The dataset has been obtained from http://ai.stanford.edu/ amaas/data/sentiment/

2. Document Representation

I tried five different document representations for the classification task:

1. Bag Of Words
2. Normalized Term Frequency Representation
3. Tf-Idf Representation
4. Unweighted average of Word2Vec word vectors (pretrained)
5. Unweighted average of GLoVE word vectors (pretrained)

Instead of training, I have obtained pretrained Word2Vec vectors from the GoogleNews word2vec vectors, and pretrained GLoVE vectors from the Common Crawl vectors with 840B tokens, 2.2M vocab and 300 dimensional vectors.

For BOW, Tf-Idf and Normalized Tf, I have used default methods available in the Scikit-Learn library for python.

3. Classification Algorithms

I have implemented 5 classification algorithms as suggested in the problem statement, namely:

1. NaiveBayes
   (a) For Tf-Idf and Normalized-Tf representations, I have used Multinoulli Distribution, where as for the other representations, I have used Bernoulli Distribution as the Likelihood Model.
   (b) This is trained using Mini-Batch Gradient Descent.
2. Logistic Regression
   (a) Performed simple Logistic regression using log-loss.
   (b) This is trained using Mini-Batch Gradient Descent.
3. Support Vector Machine

(a) Performed simple SVM Classification using hinge-loss.
(b) This is trained using Mini-Batch Gradient Descent.

4. Feedforward Neural Network
5. Performed classification using a Multi-Layer Perceptron with two hidden layers of sizes 100 each, and a 'tanh' activation function.
6. This is trained using Mini-Batch Gradient Descent as well.
7. Recurrent Neural Network (LSTM) (Two Variants)
   (a) The first variant (LSTM) uses the pre-trained word vectors, and the sequence of words is feeded into the LSTM for classification. The sequence size is taken to be 50. For the second variant (E-LSTM), the word vectors are feeded into an Embedding Layer, and the Embedding Layer is set to be trainable to optimize the word-vectors in order to perform better for the classification task.
   (b) The second variant, as expected gives better results.

I have trained all the above classification algorithms using different document representations, and computing the accuracy of each for the classification task.

4. Accuracy Table

For determining the accuracy of each combination, I have used 2500 entried of the test set, with the accuracy being 1 - (0-1 loss) in terms of percentage. This helps us in comparing the different classification algorithms and which representation suits that algorithm the best.

Model - Representation	**BoW**	**Normalized-TF**	**Tf-Idf**	**Word2Vec**	**GLoVE**
NaiveBayes	85.66	71.24	71.70	82.24	83.22
Logistic	86.24	74.30	80.58	82.52	83.72
SVM	85.62	68.94	50.88	82.58	83.88
Feedforward NN	85.60	88.10	87.18	83.54	84.18
LSTM	—	—	—	77.76	81.80
E-LSTM	—	—	—	78.38	79.12

From the table, it is evident that FeedForward Neural Network performs well for Normalized-Tf and Tf-Idf representations, whereas BoW, GLoVE and Word2Vec are well suited for all algorithms almost equally well.

For LSTM, there is lower accuracy for word2vec due to the lower overlap between the pretrained vectors and the dataset vocabulary, and the fact that the vectors are trained for a different corpora.

5. Files Description

The main files in my assignment folder are described below:

1. main.py	Contains the main wrapper code for running the various classification algorithms in different combinations of the representations.
2. representation.py	Contains the code for deriving different document representations.
3. model.py	Contains the code for different classification algorithms.

6. Running the Code

First change the directory into the assignment directory

```
cd /path/to/dir
```

If running for the first time, download the IMDB dataset into a data folder and download the glove-vectors as well as word2vec vectors into the data folder. Then run the preprocessing file as follows

```
python preprocessing.py
```

I have also added the pre-trained filtered word-vectors here:

GLoVE Vectors : https://cse.iitk.ac.in/users/guggu/glove.txt

Word2Vec Vectors : https://cse.iitk.ac.in/users/guggu/word2vec.txt

Now, we are ready to train and evaluate the models. In order to start the training, run the main.py file as follows

```
python main.py
```

The complete script will take 4 hours to complete. The complete output will be recorded in the stdout itself.