CS671 : Assignment-2
Abhishek Verma
Roll No. - 14026

Preprocessing:

- Some of the text files contained html tag
 which needed to be removed. I wrote a python script <src/preprocess.py> which reads all the files(train+test) one by one and deletes all the occurrences of html tag
 from the files.
- There is no need to do stop-word removal or stemming at this step as the <scikit> library has methods which can do this.

Helper Classes(src/util.py):

Data: This class reads all the data(train+test) and its function <getData()> returns a tuple(train_corpus, test_corpus, Y_train, Y_test). Here, train_corpus and test_corpus is a list of strings(a single review file is considered a single string). Y_train and Y_test is a list of integers(0 and 1) where 0 indicates negative label and 1 indicates positive label. This class is used by <BBoW, Tf, Tf-Idf>.
dimension of train_corpus, test_corpus: (12500) [#reviews]
dimension of Y_train, Y_test: (12500) [#reviews]

NewData: This class is same as the class <Data> except that here, train_corpus and test_corpus is a list of list of strings(each string is a single word of a review file). This class is used by <GloVe, Word2Vec>.
dimension of train_corpus, test_corpus: (12500x#words in respective review)
dimension of Y_train, Y_test: (12500) [#reviews]

MeanEmbeddingVectorizer: This class takes a python dict(key: words, value: corresponding vector) and train_corpus/test_corpus as input and transforms the train_corpus/test_corpus to a numpy array of dimension(12500 x feature size of each word). For every string(which is a review itself) in the corpus, it returns mean of vectors of corresponding words present in the string.

TfidfEmbeddingVectorizer: This is same as <MeanEmbeddingVectorizer> except that in this case, mean of vectors of every word is taken after multiplying the word with its tf-idf score.

Implementation Details and Approach:

- **Feature representation**

BBoW: To convert train_corpus and test_corpus to Binary BoW(BBoW), I used the <CountVectorizer(binary=True, stop_words='english')> method of the library module <sklearn.feature_extraction.text>.

Tf: To convert train_corpus and test_corpus to tf vector, I used the <TfidfVectorizer(stop_words='english', use_idf=False)> method of the library module <sklearn.feature_extraction.text>.

Tf-Idf: To convert train_corpus and test_corpus to tf-idf vector, I used the <TfidfVectorizer(stop_words='english')> method of the library module <sklearn.feature_extraction.text>.

GloVe: For this, I have used pre trained word vector representations available on Stanford website. This has two parts:

i. ***MeanEmbeddingVector:*** When train_corpus/test_corpus along with word vector representation is given as input to <MeanEmbeddingVectorizer> class(implemented in <src/util.py>), this class transforms the corpus to feature vectors by a mechanism discussed in previous section.

ii. ***TfidfEmbeddingVectorizer:*** When train_corpus/test_corpus along with word vector representation is given as input to <TfidfEmbeddingVectorizer> class(implemented in <src/util.py>), this class transforms the corpus to feature vectors by a mechanism discussed in previous section.

Word2Vec: I have trained this model on the given dataset(Stanford movie review) using <Word2Vec> method of <gensim.models.word2vec> python module. This method takes a copus(train_corpus/test_corpus) as input and returns a dictionary containing feature vectors of the words as in the case of GloVe discussed above. The remaining mechanism is same as that of GloVe.

- ## Classification

Logistic_regression: I obtain the feature vector for every movie_review from (BBoW, Tf, Tf-Idf, GloVe, Word2Vec) and then

i. at the time of training: use the method <LogisticRegression()> available in the python module <sklearn.linear_model> to train the model

ii. at the time of test: use the above trained model to predict the class(0 or 1).

Naive_Bayes: I obtain the feature vector for every movie_review from (BBoW, Tf, Tf-Idf, GloVe, Word2Vec) and then

i. at the time of training: use the method <BernoulliNB()> available in the python module <sklearn.naive_bayes> to train the model

ii. at the time of test: use the above trained model to predict the class(0 or 1).

Perceptron: I obtain the feature vector for every movie_review from (BBoW, Tf, Tf-Idf, GloVe, Word2Vec) and then

i. at the time of training: use the method <MLPClassifier()> available in the python module <sklearn.neural_network> to train the model

ii. at the time of test: use the above trained model to predict the class(0 or 1).

SVM: I obtain the feature vector for every movie_review from (BBoW, Tf, Tf-Idf, GloVe, Word2Vec) and then

i. at the time of training: use the method <SVC()> available in the python module <sklearn.svm> to train the model

ii. at the time of test: use the above trained model to predict the class(0 or 1).

Note: For feeding the feature representations to classifiers, I used the <Pipeline()> method of the python module <sklearn.pipeline>. This method makes training faster.

Results:

Accuracy table

	Logistic_regression	**Naive_Bayes**	**Perceptron**	**SVM**
BBoW	0.86484	0.81496	0.85284	0.74536
Tf	0.8736	0.81496	0.84372	0.6522
Tf-Idf	0.88012	0.81496	0.84168	0.62924
GloVe	0.6306(mean) 0.63516(Tf-Idf mean)	0.56964(mean) 0.56776(Tf-Idf mean)	0.62144(mean) 0.62992(Tf-Idf mean)	0.53612(mean) 0.54992(Tf-Idf mean)
Word2Vec	0.59296(mean) 0.60748(Tf-Idf mean)	0.54088(mean) 0.5452(Tf-Idf mean)	0.58092(mean) 0.5996(Tf-Idf mean)	0.54728(mean) 0.5562(Tf-Idf mean)

Note: In the above table, mean stands for **MeanEmbedding** and Tf-Idf mean stands for **TfidfEmbedding**.

Files Submitted:

- <**.py files**>**:** These are the code files.
- <**results.txt**>: This contains the output produced by <.py> files.

References:

1. http://nadbordrozd.github.io/blog/2016/05/20/text-classification-with-word2vec/