CS671A

ASSIGNMENT 2

ANUPRIY (150121)

Guidelines for running the code :

1. Create a folder named “asgn2”.
   Command - $ mkdir asgn2
2. Change directory to the folder.
   Command - $ cd asgn2
3. Unzip the “aclImdb_v1.tar.gz” dataset in this folder.
   Command - $ tar -xvzf <Path_to_zipfile>/aclImdb_v1.tar.gz
4. Unzip the submission file “150121.zip” in the folder.
   Command - $ unzip <Path_to_submission_file>/150121.zip
5. This is it. Now, each of the python scripts can be run without any error.

Structure of the submission :

1. **Data Generation Python scripts :** These are the scripts which generates the training and test data in the forms give in assignment viz. BBoW, NormTf, TfIdf, AvgWord2Vec, AvgWord2Vec weighted with TfIdf values, AvgWord2Vec using Glove, AvgWord2Vec using Glove weighted with corresponding TfIdf values, Paragraph Vectors. (Apparently 1st part of the assignment)
   a. “gen.py” : This script generates the training and testing data in BBoW (labeledBbow.feat), Normalized Tf (labeledNormTf.feat) and TfIdf (labeledTfIdf.feat) format using BoW (labeledBow.feat) format given in aclImdb dataset. These data formats are stored automatically in folder “asgn2/aclImdb/train/” and “asgn2/aclImdb/test/” respectively for training and testing data.
      Command - $ python gen.py # for creating training and test dataset
   b. “genAvgWord2Vec.py” : This script generates the feature vector of each document as average of the each word2vec vectors of the words occurring in the document (labeledAvgWord2Vec.feat). Here, word2vec model is trained using script “word2vec.py”.
      Command - $ python genAvgWord2Vec.py # for creating training and test dataset
   c. “genAvgTfIdfWord2Vec.py” : It is the same as the above but each word vector is multiplied by its tfidf weights using the tfidf representation of the document (“labeledTfIdf.feat”) generated earlier (labeledAvgTfIdfWord2Vec.feat). It also uses the same word2vec model.
      Command - $ python genAvgTfIdfWord2Vec.py # for creating training and test dataset

d. "genAvgGlove.py" : This script uses the pre-trained Glove model downloaded from here. It's 100-d model is used for training and testing purposes. This model also has to be put in the asgn2/ folder. It is the same as "genAvgWord2Vec.py" but word2vec model is replaced by this pre-trained model. (labeledAvgGlove.feat)
   Command - $ python genAvgGlove.py # for creating training and test dataset
e. "genAvgTfIdfGlove.py" : It is same as the above but each glove word vector is multiplied with its tfidf values. (labeledAvgTfIdfGlove.feat)
   Command - $ python genAvgTfIdfGlove.py # for creating training and test dataset
f. "genParaVec.py" : This uses the Doc2Vec model trained using the script "doc2vec.py". It generates paragraph vectors for each document (reviews) as its feature vectors (labeledParaVec.feat)
   Command - $ python genParaVec.py # for creating training and test dataset

2. **Classifier scripts :** These are the scripts which classifies the given test data as positive or negative review after being trained on training data (which we've obtained from previous step). (Apparently 2nd part of the assignment)
   a. There is only one script for the classification which include classifiers namely, Multinomial NB, Logistic Regression, SVM, FeedForward Neural network.
   b. You just have to follow the screen queries to run the script "classifier.py"
      Command - $ python classifier.py
   c. First script will ask for the document representation among the given representations. Then, it will ask for the classifier among the possible classifiers. Then, script will train accordingly and print the accuracy.
3. **Training of the word vectors and paragraph vectors model :**
   a. Word2Vec : This model is trained using the script "word2vec.py" with the help of gensim library (CBOW algorithm). Dimension of word vector is 300.
      Command - $ python word2vec.py
   b. Glove : This model is downloaded as pre-trained and then converted to usable word2vec format using gensim library. Dimension of Glove vector is 100.
   c. Paragraph Vectors : This model is trained using the script "doc2vec.py" again using gensim library (PV-DM algorithm). Dimension of Paragraph vector is 150.
      Command - $ python doc2vec.py

Approach :

1. **Classification :**
   a. For the classification algorithms, sklearn library is used extensively. Sklearn has classifiers for all the four classifiers that I used. To improve the accuracy, some tuning was done of the hyperparameters like hidden layers shape, learning rate, classifier subtypes, loss functions etc. Code is in "classifier.py"
   b. I've also implemented Feedforward neural network in Tensorflow apart from that of sklearn. Code is in "feedforward.py".

2. **Generation of representations :**
    a. Binary Bag of Word : It was straightforward by using the functions of csr_matrix. We get the input of “labeledBow.feat” in libSVM format whose feature vectors are in csr_matrix.
    b. Normalized Term Frequency : Apparent from code. (Discussed in class)
    c. Tf Idf : Apparent from code. (Discussed in class)
    d. Average of Word2Vec vectors in a document : In this, first word2vec model is trained using algorithm CBOW with the help of gensim library. Only training data is used for training. Now, average of each document is calculated using Bag of Words representation.
    e. Average of Word2Vec vectors weighted with Tfldf values : Same as above but weighting is done using Tfldf representation generated earlier.
    f. Average of Glove vectors : Used pre-trained model and rest is similar to (d).
    g. Average of Glove vectors weighted with Tfldf values : Used pre-trained model and rest is similar to (e).
    h. Paragraph vectors : In this, first doc2vec model is trained using the algorithm PV-DM with the help of gensim library. For training the model, each document was word tokenized separately and each document was tagged for their identities. Both train and test text data is used as corpus. Finally, each document was assigned a paragraph vector of size 150 and stored which is further used for classification.

Results :

	Bbow	Bow	NormTf	Tfldf
Multinomial NB	82.99%	81.36%	82.47%	84.52%
Logistic Regression	86.97%	86.77%	73.12%	86.29%
SVM	66.94%	73.28%	63.16%	71.88%
Feed Forward NN	86.03%	86.83%	83.35%	86.29%

	AvgWord2Vec	AvgTfIdfWord2Vec	AvgGlove	AvgTfIdfGlove	ParaVec
Logistic Regression	81.88%	68.63%	79.77%	65.50%	58.83%
SVM	78.55%	50.00%	76.79%	50.00%	60.44%
Feed Forward NN	82.11%	77.03%	80.36%	72.47%	61.44%

Hidden Layers : Bbow - **(50, 2)**, Bow - **(5, 2)**, NormTf - **(80, 4)**, Tfldf - **(50, 2)**, AvgWord2Vec - **(80, 6)**, AvgTfIdfWord2Vec - **(80, 4)**, AvgGlove - **(80, 6)**, AvgTfIdfGlove - **(20, 8)**, ParaVec - **(80, 4)**