Assignment2 NLP

Mohd Abbas Zaidi

23 March 2018

1 Results and Implementation details

1.1 Accuracy in percentage

All the options given in the Assignment were tried and almost all the permissible combinations were executed.

Classifier/ Representation	Naive bayes (Multinomial)	Naive bayes (Gaussian)	Logistic Regression	Linear SVM	Neural Net
BBOW	82.99	N/A	86.97	85.27	86.23
Normalized TF	82.47	N/A	73.12	81.16	80.37
TFIDF	83.07	N/A	88.30	87.93	88.04
Word2Vec vectors	N/A	70.99	85.14	86.42	86.44
Word2Vec Vectors with TFIDF	N/A	74.46	84.73	85.35	84.28
Glove Vectors	N/A	65.66	75.45	75.85	75.65
Glove Vectors with TFIDF	N/A	69.07	74.08	74.12	74.90
Sent2Vec	N/A	50.78	63.10	63.02	66.43
Doc2Vec	N/A	51.91	55.53	55.5	60.76

Accuracy	LSTM
Word2Vec vectors	80.10
Glove Vectors	74.22

1.1.1 Comments

- **SVM** and **Logistic Regression** give good results. Since the data consists of reviews with strongly positive or negative responses($\geq$ 7 & $\leq$ 3 out of 10), we can guess that the resultant representations are somewhat linearly separable. The Neural Network and other computationally high techniques may be overfitting, due to which the test accuracy is not very high for them

1.2 Details of Methods employed

The code contains a control functions which takes 2 numbers among it's inputs. The first represents the **Word Representation** method to be used and second the **classification** method.

1.2.1 Data Preprocessing

- The `glob` library of python was used to store each review in a list.The binary labels were assigned to the documents based on positive or negative reviews. This data was used to generate word embeddings for `Word2Vec` and `Glove` vectors.

1.2.2 Document Representations

- **BOW-** Bag of Words stored in `labeledBow.feat` file was obtained. It is stored in sparse matrix format, binary labels were given based on positive or negative reviews.
- **BBOW-** obtained from the BOW matrix by making the features binary maintaining the sparse matrix format.
- **Normalised Term Frequency** was obtained from BOW, by normalising the rows(maintaining the sparse matrix format)
- **Tfidf** was obtained by using the inbuilt `TfidfTransformer` from the `sklearn` function of python
- **Word2Vec-** Word vectors of size 300 were obtained using `gensim` library from the pretrained Google Vectors and then averaging out to find the representation for the document.
- **Word2Vec with tfidf-** the tfidf weights were use to find a weighted average of W2V vectors for each document.
- **Glove-** Glove vectors of size 50 were obtained using `gensim` from the pretrained Stanford Vectors and then averaged out.
- **Glove with tfidf-** the tfidf weights were use to find a weighted average of Glove Word Vectors.

- **Doc2Vec-** the `Doc2Vec` module in `gensim` library of python was used, documents were represented as a list of words tagged by a unique tag(used later to access their embedding). The set of unsupervised reviews was also used to train the paragraph vectors.
- **Sent2Vec-** same as Doc2Vec, just that each sentence was treated as a different document.

1.2.3 Classification Methods

- **Naive Bayes-** Multinomial and Gaussian Naive Bayes were used from the sklearn library of python.
- **Logistic Regression** and **Linear SVM** were employed using `LogisticRegression` (from `linear_model` module) and `svm.linear` module of the sklearn library of python repsectively.
- **Neural Network** was applied using `MLPClassifier`(from `neural_network` module) of the sklearn library of python. 2 hidden layers, each of size 30 were used.
- **LSTM-** LSTM was employed using `Sequential` model, `Dense` and `LSTM` layers from **keras**. In LSTM, document vectors consisted of queued word vectors(the sequence for LSTM), which were zero padded to ensure same size for all vectors. The network consisted of 1 LSTM layer(output size 5) and 1 dense layer(output layer, with 1 output). Number of epochs were varied between 5 to 10.

1.2.4 Miscellaneous

- Words not present in model vocabulary in `Word2Vec` and `Glove` were ignored.
- `nltk` was used for sentence and word tokenization.
- The representations were stored to speed up the process.