Revanth Akella
SynapseFI Kaggle Challenge
July 5, 2019

Sentiment Analysis

This report covers the approach taken for sentiment analysis of movie reviews. The report is divided into the following sections:

1. Dataset
2. Data Exploration
3. Classification and Sentiment Labeling

“Proin metus urna porta non, tincidunt ornare. Class aptent taciti sociosqu ad per inceptos hamenaeos.”

-Leo Praesen

Dataset:

The dataset is comprised of tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. Each Sentence has been parsed into many phrases by the Stanford parser. Each phrase has a PhraseId. Each sentence has a SentenceId. Phrases that are repeated (such as short/common words) are only included once in the data.

- train.tsv contains the phrases and their associated sentiment labels. We have additionally provided a SentenceId so that you can track which phrases belong to a single sentence.
- test.tsv contains just phrases. You must assign a sentiment label to each phrase.

The sentiment labels are:

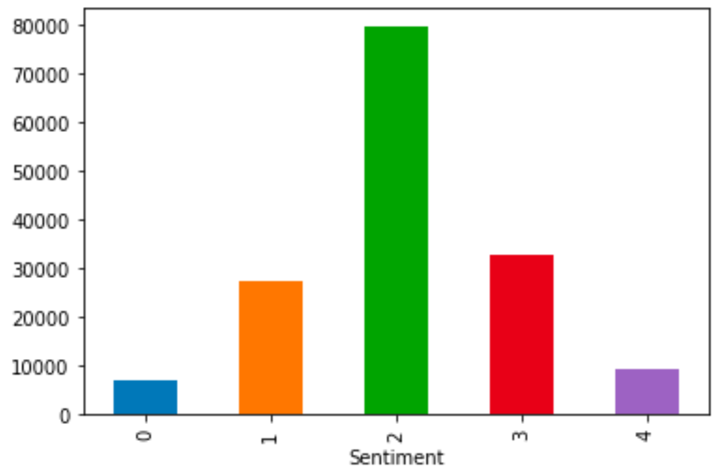
0 - negative
1 - somewhat negative
2 - neutral
3 - somewhat positive
4 - positive

Data Exploration:

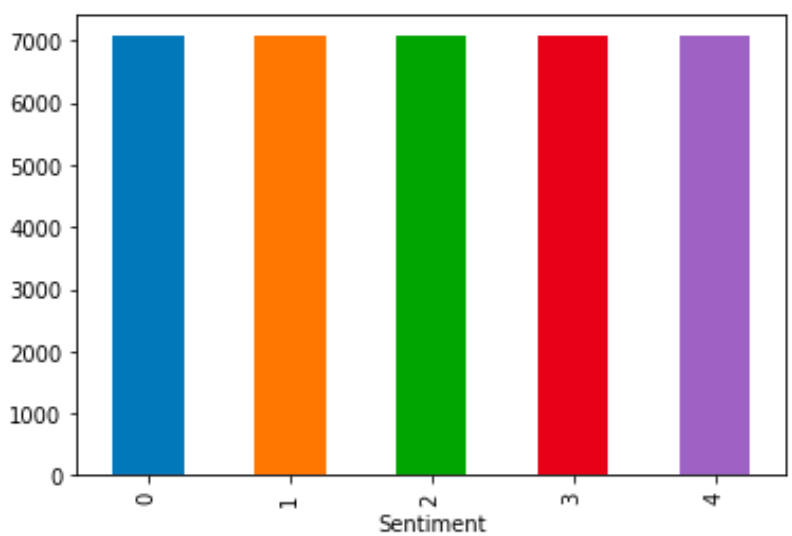
Data exploration in sentiment analysis involves examining the dataset for the following features:

1. Visualizing the distribution of the dataset:

The dataset distribution is shown in the figure below. It can be observed that the dataset is not balanced and some classes have more values than the others.

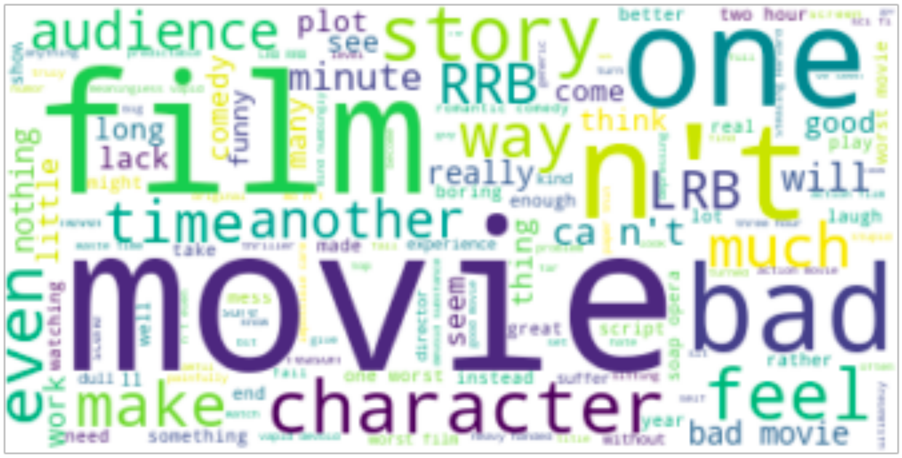


2. To balance the dataset we calculate the size of the smallest class and make sure that every class has the same number of samples. As the smallest class has 7072 samples, we prune the dataset by selecting 7072 samples from each class. The data distribution after sampling is as follows:

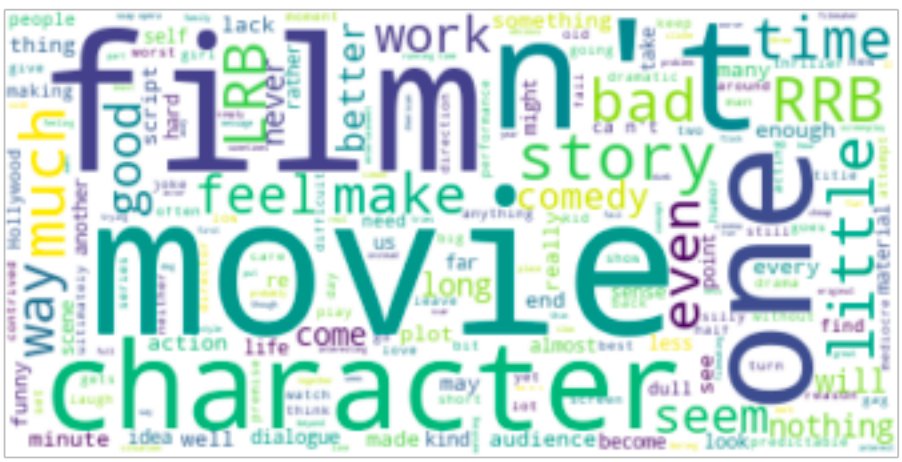


3. The word cloud distribution of the each class is as follows:

Class 0:



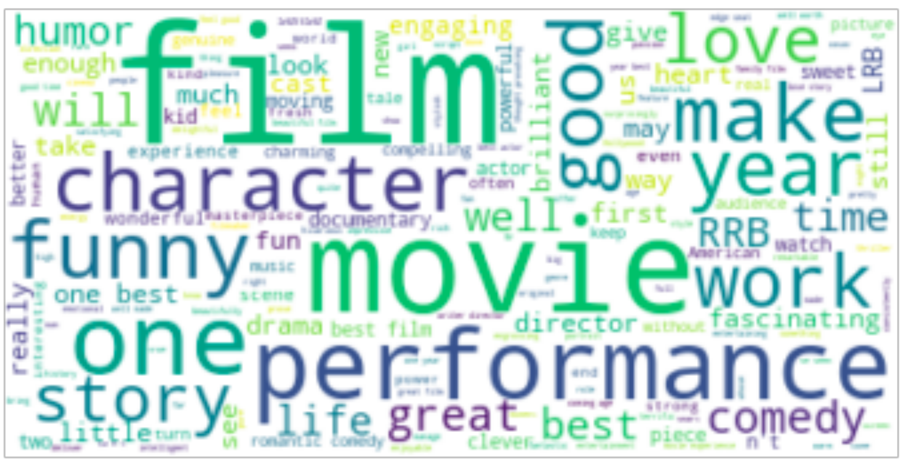
Class 1:



Class 2:

Class 3:

Class 4:



The 5 classes have “movie”, “one” and “movie” as common words that are repeated a lot. We add these words to the list of stop words.

The training set has 12212 unique words and the maximum review size is 48.

Classification and Sentiment Labeling:

For the classification models we start with Machine Learning Approaches and then move on to deep learning methods. The classification approaches are as follows:

1. Tf-IDf based classification:
   We use Multinomial Naive Bayes and Random Forest Classifiers. The input variable is the Term-Frequency Inverse Document Frequency scores.

The classification reports are as follows:

Multinomial Naive Bayes:

```
accuracy 0.47219079939668174
              precision    recall  f1-score   support

           0       0.53      0.71      0.61      2122
           1       0.43      0.35      0.38      2185
           2       0.46      0.22      0.29      2127
           3       0.40      0.37      0.38      2120
           4       0.49      0.73      0.59      2054

   micro avg       0.47      0.47      0.47     10608
   macro avg       0.46      0.47      0.45     10608
weighted avg       0.46      0.47      0.45     10608
```

Random Forest Classifier:

```
accuracy 0.5299773755656109
              precision    recall  f1-score   support

           0       0.66      0.63      0.64      2122
           1       0.46      0.35      0.40      2185
           2       0.48      0.66      0.55      2127
           3       0.43      0.38      0.40      2120
           4       0.63      0.64      0.63      2054

   micro avg       0.53      0.53      0.53     10608
   macro avg       0.53      0.53      0.53     10608
weighted avg       0.53      0.53      0.52     10608
```

2. Word2Vec models using the Google News Vectors:

The GoogleNews-vectors through Word2Vec makes use of a pre-trained model that assigns word vectors to the words. These are passed as input to the classifier.

The words are tokenized and then passed through a word vectorizer that uses the pretrained model of GoogleNews vectors.

The classifiers used here are:
Logistic Regression Classifier:

accuracy 0.5258295625942685

	precision	recall	f1-score	support
0	0.55	0.66	0.60	2122
1	0.45	0.31	0.37	2185
2	0.54	0.63	0.58	2127
3	0.44	0.33	0.38	2120
4	0.58	0.72	0.64	2054
micro avg	0.53	0.53	0.53	10608
macro avg	0.51	0.53	0.51	10608
weighted avg	0.51	0.53	0.51	10608

Random Forest Classifier:

accuracy 0.4671945701357466

	precision	recall	f1-score	support
0	0.53	0.61	0.57	2122
1	0.38	0.37	0.37	2185
2	0.46	0.48	0.47	2127
3	0.37	0.31	0.34	2120
4	0.59	0.56	0.58	2054
micro avg	0.47	0.47	0.47	10608
macro avg	0.46	0.47	0.47	10608
weighted avg	0.46	0.47	0.46	10608

3. Doc2Vec to classify the data at a document level using DBOW (Distributed Bag of Words)

Logistic Regression Classifier:

accuracy 0.4549396681749623

	precision	recall	f1-score	support
0	0.52	0.60	0.56	2103
1	0.35	0.23	0.27	2143
2	0.41	0.57	0.48	2102
3	0.37	0.25	0.30	2112
4	0.55	0.63	0.59	2148
micro avg	0.45	0.45	0.45	10608
macro avg	0.44	0.46	0.44	10608
weighted avg	0.44	0.45	0.44	10608

Random Forest Classifier:

accuracy 0.4549396681749623

	precision	recall	f1-score	support
0	0.52	0.60	0.56	2103
1	0.35	0.23	0.27	2143
2	0.41	0.57	0.48	2102
3	0.37	0.25	0.30	2112
4	0.55	0.63	0.59	2148
micro avg	0.45	0.45	0.45	10608
macro avg	0.44	0.46	0.44	10608
weighted avg	0.44	0.45	0.44	10608

It is observed that classifying at a document level is not too effective.

4. Deep Learning models:

The advantage of using Deep Learning is that we can use models that can capture more word combinations than n-gram models and the use of RNNs and LSTMs allow capturing sequential information. The models primarily follow the following preprocessing steps:

1. Tokenization
2. Embedding the tokens into an embedding matrix
3. Train the model
4. Make predictions

LSTM Model

The LSTM model is described as follows:

```
Model: "sequential_24"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_7 (Embedding)      (None, None, 100)         1374000
_________________________________________________________________
lstm_13 (LSTM)               (None, None, 64)          42240
_________________________________________________________________
lstm_14 (LSTM)               (None, 32)                12416
_________________________________________________________________
dense_68 (Dense)             (None, 5)                 165
=================================================================
Total params: 1,428,821
Trainable params: 1,428,821
Non-trainable params: 0
_________________________________________________________________
```

The results have the following accuracy: 66.02

CNN model

The CNN models is as follows:

```
Model: "sequential_29"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_12 (Embedding)     (None, 48, 100)           1374000
_________________________________________________________________
dropout_33 (Dropout)         (None, 48, 100)           0
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 48, 64)            19264
_________________________________________________________________
global_max_pooling1d_5 (Glob (None, 64)                0
_________________________________________________________________
dense_77 (Dense)             (None, 128)               8320
_________________________________________________________________
dropout_34 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_78 (Dense)             (None, 5)                 645
=================================================================
Total params: 1,402,229
Trainable params: 1,402,229
Non-trainable params: 0
_________________________________________________________________
```

The accuracy is as follows: 66.84

CNN+CRU model

The CNN+GRU model is as follows:

```
Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_13 (Embedding)     (None, 48, 100)           1374000
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 48, 64)            19264
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 24, 64)            0
_________________________________________________________________
dropout_35 (Dropout)         (None, 24, 64)            0
_________________________________________________________________
gru_1 (GRU)                  (None, 24, 128)           74112
_________________________________________________________________
dropout_36 (Dropout)         (None, 24, 128)           0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3072)              0
_________________________________________________________________
dense_79 (Dense)             (None, 128)               393344
_________________________________________________________________
dropout_37 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_80 (Dense)             (None, 5)                 645
=================================================================
Total params: 1,861,365
Trainable params: 1,861,365
Non-trainable params: 0
_________________________________________________________________
```

The accuracy is as follows: 67.08

Bi-Directional GRU model

The bidirectional GRU model is as follows:

```
Model: "sequential_31"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_14 (Embedding)     (None, 48, 100)           1374000   
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 48, 100)           0         
_________________________________________________________________
bidirectional_1 (Bidirection (None, 256)               175872    
_________________________________________________________________
dropout_38 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_81 (Dense)             (None, 5)                 1285      
=================================================================
Total params: 1,551,157
Trainable params: 1,551,157
Non-trainable params: 0
_________________________________________________________________
```

The accuracy is as follows: 66.86

Using GloVe word embeddings:

In this model we use pre-trained word embeddings. The pre-trained model here is called GloVe - Global Vectors for Word Representation and it allows us to have the embedding matrix with pre-trained weights for the words.

The model used is as follows:

```
Model: "sequential_34"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_16 (Embedding)     (None, 48, 100)           1374000
_________________________________________________________________
spatial_dropout1d_3 (Spatial (None, 48, 100)           0
_________________________________________________________________
bidirectional_4 (Bidirection (None, 48, 256)           175872
_________________________________________________________________
bidirectional_5 (Bidirection (None, 128)               123264
_________________________________________________________________
dropout_40 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_83 (Dense)             (None, 5)                 645
=================================================================
Total params: 1,673,781
Trainable params: 1,673,781
Non-trainable params: 0
_________________________________________________________________
```

The accuracy is as follows: 68.49