IE7860: Intelligent Analytics

Assignment 8 : Machine Learning for Unstructured Text

Rotten Tomatoes Movie Review Document

Overview

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee. In their work on sentiment treebanks, they used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus. This report will go on to highlight the pros and cons of various models all sentimental analysis. I have used machine learning models such as SVC, logistic regression and a few others as base line.

Data Description

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. The data also has additionally provided a SentenceId so that you can track which phrases belong to a single sentence.

test.tsv contains just phrases without the labels.

	Phrase	Sentiment
0	A series of escapades demonstrating the adage ...	1
1	A series of escapades demonstrating the adage ...	2
2	A series	2
3	A	2
4	series	2
5	of escapades demonstrating the adage that what...	2
6	of	2
7	escapades demonstrating the adage that what is...	2
8	escapades	2
9	demonstrating the adage that what is good for ...	2
10	demonstrating the adage	2
11	demonstrating	2
12	the adage	2
13	the	2
14	adage	2
15	that what is good for the goose	2
16	that	2
17	what is good for the goose	2
18	what	2
19	is good for the goose	2
20	is	2
21	good for the goose	3
22	good	3
23	for the goose	2
24	for	2
25	the goose	2
26	goose	2
27	is also good for the gander , some of which oc...	2
28	is also good for the gander , some of which oc...	2
29	is also	2

The sentiment labels are:

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

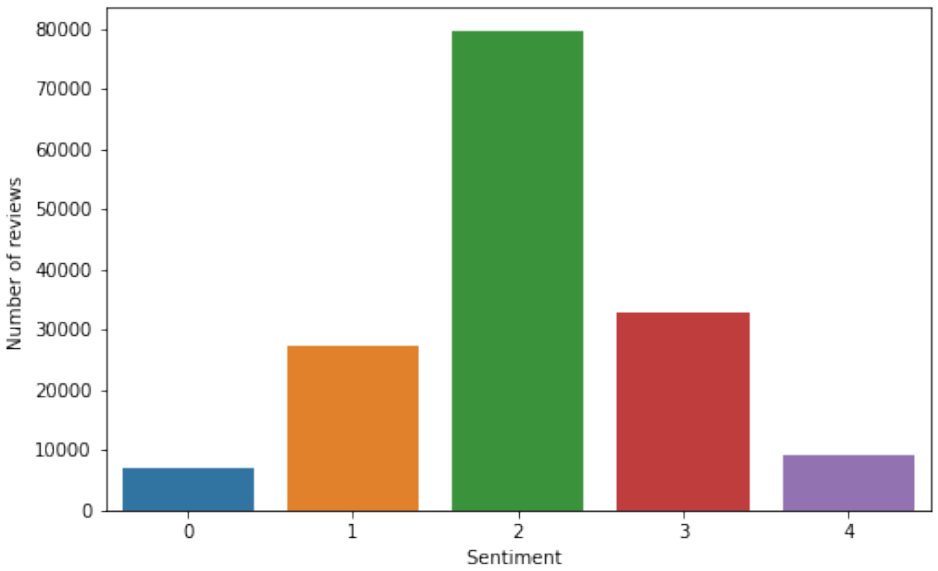
Data Pre-Processing and EDA

There wasn't much of data processing required since the date I was already free passes for a kaggle competition. However there were certain sentence with phrases with just single words. These were non-contributing phrases and would not be understanding movie reviews in terms of sentiment value. Hence this sentence ideas were identified by word length and deleted. Any sentence ID with less than two words was deleted. The following image shows the count after the deletion value and the number of data points respectively. The second image highlights the count of sentences in train and test data. It also highlights the number of words in the sentences.

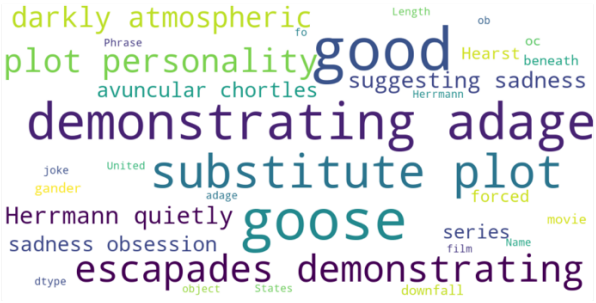
```
2    79582
3    32927
1    27273
4     9206
0     7072
```

```
#sentences in train: 8529
#sentences in test: 3310
mean words/sentence in train: 18.297572986282095
mean words/sentence in test: 20.027794561933536
```

Moving on to the exploratory data analysis, the image below highlights the distribution of the data, the key idea is to note here is that most of the reviews are neutral people tend to provide only somewhat negative or somewhat positive reviews. Extremely negative and extremely positive are very few, this shows a very strong data bias towards neutral reviews. Due to the lack of sample space for class 0 and class 4, there might be problems in classifying them with high accuracy.

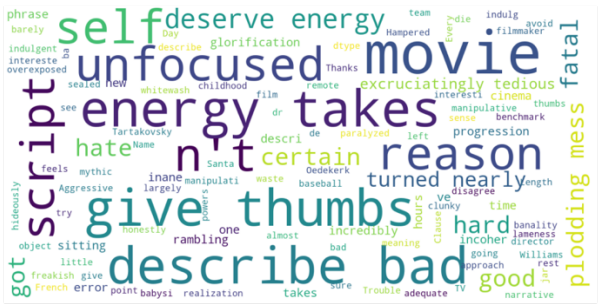


The following cluster of wordclouds show the most prominent words in each of the classes. The first image is the word cloud for the common words in both training and testing data clear words such as good, demonstrating, adage, substitute and goose are prominent. Similarly we can see the various words that are prominent in each of the classes. One abnormality to notice in the letter said is that the words that are more prominent may not highlight the actual sentiment of each of the reviews and hence cannot be used as a deciding factor for if the dataset is classified currently originally. They also include names of the movies that have the most number of reviews shows that certain movies have received more positive reviews than negative on some were positive then somewhat negative respectively.

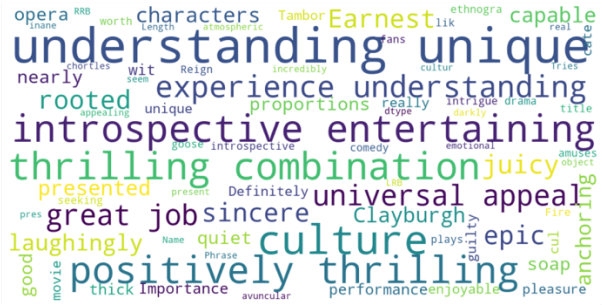


Common Words

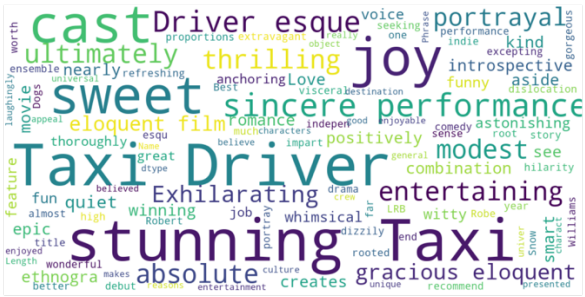
Neutral Reviews



Somewhat Negative Reviews



Negative Reviews



Somewhat Positive Reviews

Postive Reviews

Baseline Models and Evaluation

Logistic Regression

	precision	recall	f1-score	support
0	0.19	0.64	0.29	410
1	0.34	0.56	0.42	3257
2	0.91	0.65	0.76	22307
3	0.43	0.60	0.50	4662
4	0.21	0.66	0.31	576
accuracy			0.64	31212
macro avg	0.41	0.62	0.46	31212
weighted avg	0.76	0.64	0.67	31212

0.6360694604639241

Logistic regression is rather simple regression model it has classified the classes very inaccurately based on the F1 score, as mentioned previously the database is clearly seen here class two which is for neutral reviews has a relatively high F1 score whereas classes zero and

four have very low scores due to the lack of data points. The cumulative testing active received was 63.6% and can be considered a very weak base line model. Since the default logistic regression model built into the library was used there were certain warnings which were in regards to the number of iterations. The warnings mentioned that the iterations were stopped after certain maximum value since there model failed to converge.

SVC

	precision	recall	f1-score	support
0	0.38	0.48	0.42	1095
1	0.50	0.55	0.52	4852
2	0.82	0.73	0.77	17989
3	0.52	0.58	0.55	5902
4	0.41	0.55	0.47	1374
accuracy			0.66	31212
macro avg	0.52	0.58	0.55	31212
weighted avg	0.68	0.66	0.66	31212

0.6550685633730616

For SVC the Linear SVC default model built into the library was used. Using SVC the model was able to convert and show me the classification report above we can see that the F1 score for the classes with lack of data points has a better score. SPC was also able to identify more support for each of these classes and the bias is not that prominent. Overall an accuracy of 65.5% was achieved and this can be considered a better baseline model relative to logistic regression. For this model there are over thousand support data points for the extreme classes and around 5000 to 6000 support data points for classes one and three.

Voting Classifier - Logistic Regression(OVR)/SVC

	precision	recall	f1-score	support
0	0.38	0.48	0.43	1112
1	0.51	0.55	0.52	5010
2	0.86	0.69	0.77	20097
3	0.42	0.61	0.50	4438
4	0.20	0.67	0.31	555
accuracy			0.65	31212
macro avg	0.47	0.60	0.50	31212
weighted avg	0.71	0.65	0.67	31212

0.6471549404075355

To get a better baseline model, I went ahead and applied a voting classifier between logistic regression with one versus all as a supermodel and SVC. But since the voting classify choose the logistic regression model over the SVC model for class fourth we can see the reduced F1 score for that loss due to the lack of support data points. This also impacted the overall accuracy and hence we have a decrease in accuracy from SVC. The accuracy is not of importance much here since we can clearly see that class four is not being identified properly due to the inability of logistical question to classify the extreme two classes properly. Hence we would go on and choose SVC as the better among these three approaches.

Machine Learning Models Compared

All of the models below got built using Keras and tensorflow 2.0 with early stopping callbacks. In the case of unstructured text data all data points are very unique and the testing data and validation data only slightly differ since both are unseen by the model previously. To avoid further bias from testing data I choose to use validation data and compare results based on validation accuracy.

GRU

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 100)         1300000
_________________________________________________________________
gru (GRU)                    (None, None, 64)          31872
_________________________________________________________________
gru_1 (GRU)                  (None, 32)                9408
_________________________________________________________________
dense (Dense)                (None, 5)                 165
=================================================================
Total params: 1,341,445
Trainable params: 1,341,445
Non-trainable params: 0
_________________________________________________________________
```

I began with a simple GRU model, it contained four layers where the first embedding layer had hundred nodes, the second GRU layer at 64 nodes, the third GRU layer had 32 nodes and a final dance player had five output nodes. The loss function was used as categorical cross entropy. The total number of parameters is high since each of the words was tokenized and vectored. The vectoring was done to maintain the order of words and not consider each word as a separate feature.

```
Train on 124848 samples, validate on 31212 samples
Epoch 1/3
124848/124848 [==============================] - 98s 781us/sample - loss: 1.0274 - accuracy: 0.5940 - val_loss: 0.8686 - val_accuracy: 0.6511
Epoch 2/3
124848/124848 [==============================] - 97s 778us/sample - loss: 0.8260 - accuracy: 0.6663 - val_loss: 0.8296 - val_accuracy: 0.6660
Epoch 3/3
124848/124848 [==============================] - 110s 882us/sample - loss: 0.7663 - accuracy: 0.6889 - val_loss: 0.8206 - val_accuracy: 0.6703
CPU times: user 15min 41s, sys: 9min 23s, total: 25min 4s
Wall time: 5min 4s
```

The use of the categorical cross entropy as loss function was a mistake since it is better for multi-class problems and not multi-label problems due to which I was getting lower validation accuracy of 67%. Even though the results were significantly better than the baseline. The improvement expected was way higher. .I didn't simply changed the loss function from categorical cross entropy to binary cross entropy which is better suited for multi-label classification. Based on which we can clearly see a very big improvement in training accuracy as well as validation accuracy. Validation accuracy is at 86.92%. Such was an improvement that was expected earlier.

```
Train on 124848 samples, validate on 31212 samples
Epoch 1/3
124848/124848 [==============================] - 96s 765us/sample - loss: 0.3625 - accuracy: 0.8448 - val_loss: 0.3097 - val_accuracy: 0.8614
Epoch 2/3
124848/124848 [==============================] - 101s 809us/sample - loss: 0.2947 - accuracy: 0.8678 - val_loss: 0.2980 - val_accuracy: 0.8680
Epoch 3/3
124848/124848 [==============================] - 100s 802us/sample - loss: 0.2780 - accuracy: 0.8762 - val_loss: 0.2938 - val_accuracy: 0.8692
CPU times: user 15min 39s, sys: 8min 58s, total: 24min 38s
Wall time: 4min 56s
```

LSTM

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 100)         1300000
_________________________________________________________________
lstm (LSTM)                  (None, None, 64)          42240
_________________________________________________________________
lstm_1 (LSTM)                (None, 32)                12416
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 165
=================================================================
Total params: 1,354,821
Trainable params: 1,354,821
Non-trainable params: 0
_________________________________________________________________
```

The LSTM model used is again relatively similar to the GRU model instead of the GRU layers I have used LSTM layers of the same dimensions. The performance of LSDM was expected to be better than GRU. Due to the inherit property of LSTM to have long short term memory. The model has an embedding layer of 100 nodes and 2 LSTM layers of 64 and 32 nodes respectively. The final output layer is a dense layer with five output nodes.

```
Train on 124848 samples, validate on 31212 samples
Epoch 1/3
124848/124848 [==============================] - 131s 1ms/sample - loss: 0.4073 - accuracy: 0.8374 - val_loss: 0.3607 - val_accuracy: 0.8509
Epoch 2/3
124848/124848 [==============================] - 120s 964us/sample - loss: 0.3338 - accuracy: 0.8558 - val_loss: 0.3131 - val_accuracy: 0.8592
Epoch 3/3
124848/124848 [==============================] - 115s 919us/sample - loss: 0.2918 - accuracy: 0.8685 - val_loss: 0.3009 - val_accuracy: 0.8646
CPU times: user 18min 57s, sys: 12min 27s, total: 31min 24s
Wall time: 6min 5s
```

The accuracy for training as well as testing is very similar to the GRU model. With a validation accuracy of 86.46% and validation accuracy of 86.85%. This also is a big improvement from the baseline of 67% and can also be considered a very strong model.

Bidirectional GRU

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 50, 100)           1300000
_________________________________________________________________
spatial_dropout1d (SpatialDr (None, 50, 100)           0
_________________________________________________________________
bidirectional (Bidirectional (None, 50, 128)           63744
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                31104
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 325
=================================================================
Total params: 1,395,173
Trainable params: 1,395,173
Non-trainable params: 0
_________________________________________________________________
```

Bidirectional GRU is an upgrade from the regular GRU in terms that it can look at information from the previous time step and the later time step. Similar to G are you it also has a reset gate and an update gate at each layer. The update gate acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add. The reset gate is

another gate is used to decide how much past information to forget.

```
Train on 124848 samples, validate on 31212 samples
Epoch 1/3
124848/124848 [==============================] - 174s 1ms/sample - loss: 0.3483 - accuracy: 0.8474 - val_loss: 0.3026 - val_accuracy: 0.8605
Epoch 2/3
124848/124848 [==============================] - 173s 1ms/sample - loss: 0.2902 - accuracy: 0.8688 - val_loss: 0.2907 - val_accuracy: 0.8682
Epoch 3/3
124848/124848 [==============================] - 176s 1ms/sample - loss: 0.2749 - accuracy: 0.8761 - val_loss: 0.2880 - val_accuracy: 0.8686
CPU times: user 36min 27s, sys: 27min 20s, total: 1h 3min 48s
Wall time: 8min 42s
```

Due to the bidirectional condition the model takes almost twice as much long as a normal GRU. Without a significant increase in both training accuracy and validation accuracy. The validation accuracy achieved for this model is 86.86% at the final iteration. However this slight difference is neither significant enough or adds value to compensate for the additional training time. Since this training time will also linearly propagate for predictions and will slow down the use of this model in a practical environment. After all these models have been tested we still have the original GRU model as the most accurate and efficient.

CNN

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 50, 100)           1300000
_________________________________________________________________
dropout (Dropout)            (None, 50, 100)           0
_________________________________________________________________
conv1d (Conv1D)              (None, 50, 64)            19264
_________________________________________________________________
global_max_pooling1d (Global (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               8320
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 645
=================================================================
Total params: 1,328,229
Trainable params: 1,328,229
Non-trainable params: 0
_________________________________________________________________
```

Coming to one of the most significant models in recent time due to its efficiency and accuracy in classification problems whether it is multimedia or unstructured text. I built a simple CNN model with an initial embedding layer which is required since we have working with unstructured text data. The following layers are a drop out layer, a conv1D layer, a max pooling layer, dense layer, another dropout layer and a final dense output layer. They have (50,100), (50,100), (50,64), 64, 1228,128,5 nodes respectively.

```
Train on 124848 samples, validate on 31212 samples
Epoch 1/3
124848/124848 [==============================] - 30s 242us/sample - loss: 0.3472 - accuracy: 0.8467 - val_loss: 0.3032 - val_accuracy: 0.8629
Epoch 2/3
124848/124848 [==============================] - 29s 232us/sample - loss: 0.2825 - accuracy: 0.8718 - val_loss: 0.2903 - val_accuracy: 0.8682
Epoch 3/3
124848/124848 [==============================] - 40s 317us/sample - loss: 0.2580 - accuracy: 0.8839 - val_loss: 0.2882 - val_accuracy: 0.8708
CPU times: user 5min 23s, sys: 3min 31s, total: 8min 55s
Wall time: 1min 38s
```

As mentioned earlier CNN due to its efficiency in classification problems is very fast and produces results in fraction of the time that previous models have taken. For example the GRU model took almost 18 minutes for three epochs, while CNN only took five minutes with the load distributed over six cores and six hyper threaded cores. .We eventually end up with a relatively high accuracy of

87.08% on validation data, which is a drastic improvement from all the previous models that we have tested.

GRU-CNN

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_6 (Embedding)      (None, 50, 100)           1300000
_________________________________________________________________
dropout_5 (Dropout)          (None, 50, 100)           0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 50, 256)           176640
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 50, 32)            24608
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 32)                0
_________________________________________________________________
dense_8 (Dense)              (None, 64)                2112
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_9 (Dense)              (None, 5)                 325
=================================================================
Total params: 1,503,685
Trainable params: 1,503,685
Non-trainable params: 0
_________________________________________________________________
```

Since GRU and CNN models were the most promising results. I decided to combine the board and observe the performance unstructured text classification problem. The model shown above has an embedding layer similar to the CNN model followed by a dropout layer, by directional G are you layer, conv1D layer, global max pooling layer, dense layer, another dropout layer and a final dense output layer. The dimensions of which can be seen in the image above. By using a bidirectional GRU layer in between the convolution and Max cooling layer. There is a significant increase in model complexity.

```
Train on 124848 samples, validate on 31212 samples
Epoch 1/4
124848/124848 [==============================] - 317s 3ms/sample - loss: 0.3747 - accuracy: 0.8375 - val_loss: 0.3072 - val_accuracy: 0.8576
Epoch 2/4
124848/124848 [==============================] - 303s 2ms/sample - loss: 0.2973 - accuracy: 0.8666 - val_loss: 0.2905 - val_accuracy: 0.8669
Epoch 3/4
124848/124848 [==============================] - 292s 2ms/sample - loss: 0.2736 - accuracy: 0.8771 - val_loss: 0.2871 - val_accuracy: 0.8697
Epoch 4/4
124848/124848 [==============================] - 340s 3ms/sample - loss: 0.2586 - accuracy: 0.8840 - val_loss: 0.2881 - val_accuracy: 0.8701
CPU times: user 1h 5min 51s, sys: 1h 9min 20s, total: 2h 15min 11s
Wall time: 20min 52s
```

The early stoppage call back decided to go with 4 iterations in retrospect to 3 iterations in all the previous models. This is probably due to the significant change in the loss function value after each iteration. But due to the added complexity as mentioned before there is a huge increase in the time it takes to train the model. From the image above we can see that it takes over one hour to train for four iterations whereas the CNN model only took five minutes and the GRU model only took 18 minutes. Search and you deficiency cannot be justified by the accuracy that was achieved of 87.01%. Whereas the CNN model was able to achieve 87.08% with just five minutes of training.

Conclusion

	GRU-categorical	GRU-binary	LSTM	Bi-GRU	CNN	GRU-CNN
Train-Accuracy	0.6889	0.8762	0.8685	0.8761	0.8839	0.8840
Train-Loss	0.7663	0.278	0.2918	0.2749	0.2580	0.2586
Val-Accuracy	0.6703	0.8692	0.8646	0.8686	0.8708	0.8701
Val-Loss	0.8206	0.2938	0.3009	0.2880	0.2882	0.2881
Time	15:41	15:39	18:57	36:57	5:23	65:51

From the above table we can see that the Muslim was used by the GRUCNN network and the by directional G are you network with 65 minutes and 36 minutes respectively. The results from the GRUCNN are very strong and above most of the models that were tested. The results do not justify the computation inefficiency which will be linearly transferred to the production stage and the practical application of these models. Especially since the CNN network has a significantly better result of 87.08% for validation accuracy with computation time around 5 minutes. The earlier models were also very strong in terms of accuracy but due to the significant cost of these models they are not preferred. One another key factor to notice was when I used categorical crossentropy for the GRU model in the beginning the less value was very high which pointed me to think that there is something wrong in my approach. Hence finally we can see that CNN models are very strong for classification of multi-label unstructured text data such as the movie reviews from rotten tomatoes.