Movie Reviews, Sentiment Analysis Project

December 2019

Terrance Randolph

IST 664 Natural Language Processing

M.S Applied Data Science (Student)

tmrandol@syr.edu

Table Of Contents

Objective

Movies depict the stories of our lives in both fictional and non-fictional manners, they convey happiness, sadness, action, comedy and sparks mind to explore topics that pledges us all. Therefore, individuals have been granted the power to share their views on such art and partake in a healthy criticism. Although, that may have been the initial intentions, now-a-days the movie reviews are blanut and others unseemly but, these reviews whether on movies or products are not the arbiters of people's participation in that good or service. Therefore, leaving analysts to understand and attempt to predict how a movie will perform via Natural Language Processing of historic reviews.

The Problem

Where is such a training dataset?

How can I clean and process the data?

What is the accuracy of a good prediction model ?

Will the predictive model label new review sentiment ?

Dataset: Kaggle Rotten Tomatoes Movie Reviews

Sentiments: negative(0), somewhat negative(1), neutral(2), somewhat positive(3), positive (4)

Step 1 (Data Cleaning)

Overview:

The data must be read into the program and formatted in the correct data-type, character parsing and be homogenous. The Reviews are read in by the ‘review’ and ‘ratings’ columns which captures the movie rater text and their sentiment. Once read into the computer it must be parsed/separated by groups which are partitioned by a space between characters (tokenization). Next, those tokens are searched through and homogenized by changing each token to lowercase without discriminating against proper nouns. Once these tasks are

completed the data is ready to be analyzed or processed further, therefore, leaving a dataset free from misinterpretation due to capitalization or errors due to invalid inputs.

Step 2 (Feature Functions)

Overview :

When analyzing the Movie Reviews dataset there are a few valuable features that properly convey structure and evaluates its reactions to different forms/feature implementation. The first feature is Tokenization, which parses the corpus by words and punctuations while separating each token by a comma. Next are stopword, this particular feature filters the corpus looking for common speech pausing and/or commonly used words such as 'and', 'the', 'a' etc… however, Negation ignites the same filtering but with words of negative sentiment such as 'not', 'don't' etc… Following are Regular Expression filters which allows the extraction of tokens with symbols and non-alphabet characters from token or perhaps the inverse as well.

Moreover, Stemmers such as Porter and Lancaster (more aggressive) which removes the suffix of a token therefore, changing 'destruction' to 'destruct'. Next, capturing the Frequency Distribution words in a corpus is important for understanding the context of the text. However, in this case reviews are shorter therefore, merging reviews conveys insight into sentiment and context of majority of reviews. Next, Bigrams sifts through the tokens and find the most common pairs of tokens ordered by highest frequency (Raw) and by probability of Bigram tokens accompanying one another frequently in corpus (Pointwise Mutual Information). Lastly, the sentiment and subject clause bags-of-words are matched with each word from the dataset tokens and the pre-established dictionary of english words with sentiment paired to it. Therefore, allowing the analyst to consider the majority of token sentiment in phrases/reviews to account for tokens whos sentiment skews the data or which token is not but should skew the phrases/review sentiment.

The filters are evaluated individually and compared to the same reviews just unfiltered. Therefore, each feature has a function and each function accepts the returned corpus tokenized and lowercase. Then all feature functions are looped through and prints the same reviews with and without features (View code for details).

--- Read 156060 phrases, using 20 random phrases

S2 Reviews Sampled:

```
----- PHRASE SAMPLED -----

(['seductively'], 2)
(['engaging', 'and', 'literate'], 3)
(['for', 'most', 'movies', ',', '84', 'minutes', 'is', 'short', ','], 1)
(['louiso'], 2)
(['shoot', 'a', 'lot', 'of', 'bullets'], 2)
(['any', 'cost'], 2)
(['the', 'same', 'time', 'to', 'congratulate', 'himself', 'for', 'having', 'the', 'guts', 'to',
'confront', 'it'], 2)
(['is', 'impossible', 'to', 'find', 'the', 'film', 'anything', 'but', 'appalling', ',', 'shamelessly',
'manipulative', 'and', 'contrived', ',', 'and', 'totally', 'lacking', 'in', 'conviction'], 0)
(['i', 'spy'], 2)
(['flatula'], 1)
(['to', 'cut', 'your', 'teeth', 'in', 'the', 'film', 'industry'], 2)
(['you', 'just', 'know', 'something', 'terrible', 'is', 'going', 'to', 'happen', '.'], 2)
(['based', 'on', 'three', 'short', 'films', 'and', 'two', 'features'], 2)
(['this', 'side'], 2)
(['disney', "'s", 'rendering'], 2)
(['what', 'emerges', 'is', 'an', 'unsettling', 'picture', 'of', 'childhood', 'innocence', 'combined',
'with', 'indoctrinated', 'prejudice', '.'], 3)
(['falls', 'short', 'in', 'explaining', 'the', 'music', 'and', 'its', 'roots'], 1)
(['so', 'prevalent', 'on', 'the', 'rock'], 2)
(['moldy-oldie'], 2)
(['into', 'unhidden', 'british'], 2)
```

S2 Stemmers:

Explanation:

The Lancaster Stemmer has changed ‘accomplishment’ to ‘accompl’ and ‘surprise’ to ‘surpr’ which could assist in the matching of sentiment by word however such drastic parsing of suffix breeds more confusion for the compiler then anything else. However, the use of a stemmer is needed for many impactful words to be stripped and matched to word dictionaries for sentiment acquisition.

Implemented Results:

```
--- STEMMER LANCASTER EXAMPLE AFTER ---
Read 156060 phrases, using 20 random phrases

(["'s", 'a', 'movy', 'that', 'accompl', 'so', 'much', 'that', 'on', 'view', 'ca', "n't", 'poss', 'be',
'enough'], 4)
(['ton', 'shift'], 2)
(['with', 'which'], 2)
(['big-bug'], 2)
([',', 'but', 'for', 'their', 'look'], 2)
(['at', 'the', 'start'], 2)
(['produc'], 2)
```

```
(['she', 'unbridl', 'delight'], 4)
(['the', 'vis', 'and', 'envelop', 'sound', 'of', 'blu', 'crush', 'mak', 'thi', 'surpr', 'dec',
'flick', 'wor', 'a', 'summertim', 'look-see', '.'], 3)
(['mark', 'a', 'modest', 'if', 'enco', 'return'], 3)
(['for', 'an', 'unfocus', 'screenplay'], 1)
(['giv', 'it', 'an', 'unqual', 'recommend'], 2)
(['thi', 'enthral', 'docu', '...', 'is', 'at', 'ont', 'play', 'and', 'haunt', ',', 'an', 'in-depth',
'portrait', 'of', 'an', 'iconoclast', 'art', 'who', 'was', 'funda', 'unknow', 'ev', 'to', 'his',
'closest', 'friend', '.'], 4)
(['splashing'], 2)
(['wo', "n't", 'stand', 'the', 'cold', 'light', 'of', 'day'], 0)
(['vain'], 2)
(['reign', 'of', 'fir', 'nev', 'com', 'clos', 'to', 'recov', 'from', 'it', 'dem', 'prem', ','], 1)
(['a', 'bad', 'day', 'of', 'golf'], 2)
(['predict', 'or'], 1)
(['the', 'film', "'s", 'end'], 2)
```

S2 RegEx (Remove Non-Alpha):

Explanation:

Due to the regular expression feature function extracting non-alphanumeric character from tokens, that of which leaves most reviews untouched. However, those small number of reviews are resubmitted into the dataset removed of characters filtered by regex: ('^[^a-z]+$')

Implemented Results:

```
--- NON ALPHA EXAMPLE AFTER ---
Read 156060 phrases, using 20 random phrases

(['like', 'me', ',', 'think', 'an', 'action', 'film', 'disguised', 'as', 'a', 'war', 'tribute', 'is',
'disgusting', 'to', 'begin', 'with'], 0)
(['a', 'metaphor', 'for', 'a', 'modern-day', 'urban', 'china', 'searching', 'for', 'its', 'identity',
'.'], 3)
(['burgeoning'], 2)
(['a', 'few', 'new', 'swings'], 3)
(["'ll", 'only'], 2)
(['yu', 'clearly', 'hopes', 'to', 'camouflage', 'how', 'bad', 'his', 'movie', 'is', '.'], 0)
(['a', 'sensitive', 'and', 'astute'], 3)
(['ford', 'administration', "'s"], 2)
(['unsentimental', ',', 'straightforward', 'text'], 2)
(['us', 'intriguing', 'glimpses', 'of', 'the', 'insights', 'gleaned', 'from', 'a', 'lifetime', 'of',
'spiritual', 'inquiry'], 3)
(['dorkier', 'aspects'], 2)
(['how', 'inseparable'], 2)
(["'s", 'all', 'entertaining', 'enough', ','], 2)
(['credits', 'roll'], 2)
(['1953'], 2)
```

```
(['working', 'man'], 2)
(['by', 'its', 'awkward', 'structure', 'and', 'a', 'final', 'veering'], 1)
(['writer-actor'], 2)
(["'", 'n'], 2)
(['watch', ','], 2)
```

S2 Stop Words Filters & Negation :

Explanation:

An example of non- alphanumeric words being filtered out would be '--' which would in this instance remove the token altogether. Moreover, the first sentence 'lends the settings' has extracted 'the' also, 'a somber trip' removed the token 'a' stopword in order to capture words that will more likely than not have a sentiment of review deciding factor.

Implemented Results:

```
--- STOPWORDS NEGATION EXAMPLE AFTER ---
Read 156060 phrases, using 20 random phrases

(['lends', 'setting', 'ethereal', 'beauty', 'asian', 'landscape', 'painting'], 3)
(['somber', 'trip'], 2)
(['love', 'humility'], 3)
(['greedy'], 1)
(['four', 'weddings', 'funeral', 'bridget', 'jones', "'s", 'diary'], 2)
(['establishes', 'ominous', 'mood', 'tension', 'swiftly', ','], 2)
(['carpenter'], 2)
(['meat', 'freezers'], 2)
(['empty'], 2)
(['mind', ';', 'boobs', 'fantasti'], 2)
(['absolute', 'delight', 'audiences'], 4)
(['interest', 'almost', 'solely', 'exercise', 'gorgeous', 'visuals'], 3)
(['overlapping'], 2)
(['dramatic', 'weight'], 1)
(["'s", 'life-affirming', '--', 'vulgar', 'mean'], 0)
(['mesmerised'], 3)
(['logic'], 3)
(['affectionately', 'goofy', 'satire'], 3)
(['comic', 'premise'], 2)
(['song'], 2)
```

S2 Frequency Distribution :

Explanation:

It is evident by viewing the most common words from a sample of twenty phrases/reviews that the corpus needs to filter out stop words because ‘and’ , ‘.’, ‘a’ are filler words for the tokens that sentiments can be drawn from.

Implemented Results:

```
--- 3 MOST COMMON WORDS ---
[('and', 6), (',', 6), ('a', 5)]
```

S2 Bigrams Raw & PMI :

Explanation:

The most common pairs from the selected phrases/reviews are ‘the’ and ‘film’ which makes sense considering the reviews are about films however, the bigrams most likely to occur are ‘to’ and ‘be’ and the sampled dataset is only of twenty reviews.

Implemented Results:

```
--- PMI FREQUENCY TOP 2 PAIRS ---
[(('to', 'be'), 5.961931959166481),
 (('it', "'s"), 3.7395395378300336),
 (('of', 'the'), 1.9175378398080278)]

 --- RAW FREQUENCY TOP 2 PAIRS ---
[(('the', 'film'), 0.016666666666666666)]
```

S2 LIWC Sentiment lexicon :

Explanation:

The feature function acquired a word match from both reviews and LIWC document and prints the sentiment for each word such as ‘class’ (neutral sentiment) and other words such as ‘of’, ‘films’ and ‘women’ reports false on positive and negative sentiments which aligned with review sentiment of 2 (Neutral). That of which validates the sentiment evaluation.

Implemented Results:

```
--- LIWC SENTIMENT POS/NEG ---
Read 156060 phrases, using 5 random phrases

----- PHRASE SAMPLED -----

(['the', 'class', 'of', 'women', "'s", 'films'], 2)
(['into', 'theaters'], 2)
(['reconciled'], 2)
(['sloughs', 'one', "'s", 'way'], 1)
(['a', 'matter', 'of', 'taste'], 2)

----- PHRASE AFTER -----

     the ==> Pos-> False  ... Neg-> False

     class ==> Pos-> False  ... Neg-> False

     of ==> Pos-> False  ... Neg-> False

     women ==> Pos-> False  ... Neg-> False

     's ==> Pos-> False  ... Neg-> False
     films ==> Pos-> False  ... Neg-> False

     into ==> Pos-> False  ... Neg-> False

     theaters ==> Pos-> False  ... Neg-> False
```

S2 Subject Clause Sentiment lexicon :

Explanation:

The feature function acquired a word match from both reviews and Subject Clause document and prints the sentiment for each word. Leaving tokens such as ‘would’ as a neutral sentiment while other words such as ‘many’, ‘require’ and ‘on’ reports false on all sentiments which aligned with review sentiment of 1 (somewhat negative). That of which teeters on the border of neutral therefore, granting validity to the sentiment evaluation.

Implemented Results:

```
 --- SUBJECT CLAUSE SENTIMENT POS/Ntrl/NEG ---
Read 156060 phrases, using 5 random phrases

----- PHRASE SAMPLED -----

(['would', 'require', 'many', 'sessions', 'on', 'the', 'couch', 'of', 'dr.', 'freud'], 1)
```

```
(['as', 'a', 'director', 'washington', 'demands', 'and', 'receives', 'excellent', 'performances'], 4)
(['unrecoverable'], 0)
(['the', 'thrill'], 3)
(['to', 'terms', 'with', 'his', 'picture-perfect', 'life'], 2)

----- PHRASE AFTER -----

     would ==> Pos-> False  ... Ntrl-> True  ... Neg-> False

     require ==> Pos-> False  ... Ntrl-> False  ... Neg-> False

     many ==> Pos-> False  ... Ntrl-> False  ... Neg-> False

     sessions ==> Pos-> False  ... Ntrl-> False  ... Neg-> False

     on ==> Pos-> False  ... Ntrl-> False  ... Neg-> False

     the ==> Pos-> False  ... Ntrl-> False  ... Neg-> False

     couch ==> Pos-> False  ... Ntrl-> False  ... Neg-> False

     of ==> Pos-> False  ... Ntrl-> False  ... Neg-> False

     dr. ==> Pos-> False  ... Ntrl-> False  ... Neg-> False

     freud ==> Pos-> False  ... Ntrl-> False  ... Neg-> False
```

Step 3 (Filtered & Non-Filtered Evaluations)

Overview :

Data analysis models such as NB (Naive Bayes Classifier) which partitions the data into groups that are classified after their group name. Then, those classifications (groups) are measured individually against the total dataset and that probability is used in grouping new ungrouped data points. Therefore, once new information is introduced to an already partitioned dataset, it's surrounding data-points are grouped with the new point as the centroid (center data-point in group). That new group will include previously classified data-points and those probabilities are weighed against the amount of classified data-points in the new group. Leaving a new probable predicted classification of the new data-points. Therefore, it's implementation in the Kaggle Rotten Tomato Movie Review dataset is imperative for proper analysis and predictions of movie reviews.

Likewise, evaluating classifiers accuracy is just as important to data prediction, and the method chosen is confusion matrix accuracy via Precision and Recall. Once data is partitioned/classified it must be evaluated for accuracy and confusion matrices are a common method of comparing results. Those comparisons are between classified labeled-data verses extracted labeled-data (evaluated without data) and performing a one-to-one comparison of labels from extracted data to its predicted labels (classification). The True & False Positives & Negatives are checked for correctly predicted observations out of total positive predictions (Precision - label matching correctly) and to overall/total observations (Recall - total labels attempted)(cite). Lastly, F1 scoring which grants a weighted average to the accuracy that assists in normalizing unbalanced data (F1 Score = 2*(Recall * Precision) / (Recall + Precision)).

Moreover, evaluating the data with Precision and Recall methods then grabbing the weighted averages (F1 measures). First, I divided the data randomly into five even (in count/quantity) samples (short for folds) then performed Precision and Recall on each fold and perform F1 analysis to produce the weighted average. After each fold accuracy is measured those Folds are Cross Validated (F1 scores averaged) which is conducive to understanding and classifying the data's minute details for better predictions.

--- Filtered Feature vs. Non-Filtered Feature Evaluations

S3 Stemmers:

Explanation:

The non filtered classifier (47%) seems to improve once Lancaster Stemmer is initiated with the filtered (50%) model. Most likely due to the extraction of words is suffixes that can be removed in order to match word general word presentence while creating a homogenous bag-of-words.

Implemented Evaluation Results:

```
----- !LANCASTER STEMMER! -----

-------- Non_Filtered Classifier --------
Read 156060 phrases, using 2500 random phrases
4805
Each fold size: 500

Average Precision       Recall          F1      Per Label
0             0.119      0.218      0.153
```

```
1             0.209      0.328      0.253
2             0.829      0.605      0.699
3             0.219      0.361      0.270
4             0.115      0.181      0.137

Macro Average Precision Recall          F1      Over All Labels
              0.298      0.338      0.303

Label Counts {0: 120, 1: 450, 2: 1267, 3: 516, 4: 147}
Micro Average Precision Recall          F1      Over All Labels
              0.515      0.461      0.471

-------- Filtered Classifier --------
Read 156060 phrases, using 2500 random phrases
3545
Each fold size: 500

Average Precision       Recall          F1      Per Label
0             0.107      0.156      0.121
1             0.272      0.378      0.311
2             0.833      0.629      0.716
3             0.239      0.386      0.294
4             0.109      0.257      0.149

Macro Average Precision Recall          F1      Over All Labels
              0.312      0.361      0.318

Label Counts {0: 111, 1: 421, 2: 1307, 3: 523, 4: 138}
Micro Average Precision Recall          F1      Over All Labels
              0.542      0.494      0.502
```

S3 Regular Expressions:

Explanation:

The removal on all non alphabetic and numeric characters did not improve the 47% classifier much primarily because their are only a few non alphanumeric characters fitting the description. For instance, one review was '--rb--' and the filter expressed 'rb' which is great however, the tokens contain abstract features tend to not be commonly used lexicon. Therefore, rendering the two classifiers virtually the same.

Implemented Evaluation Results:

```
----- !NON-ALPHABET FILTER! -----

-------- Non_Filtered Classifier --------
Read 156060 phrases, using 2500 random phrases
4778
Each fold size: 500
```

```
Average Precision       Recall          F1      Per Label
0               0.098       0.140       0.110
1               0.232       0.354       0.279
2               0.840       0.606       0.703
3               0.209       0.341       0.258
4               0.092       0.267       0.136

Macro Average Precision Recall          F1      Over All Labels
                0.294       0.342       0.297

Label Counts {0: 104, 1: 442, 2: 1275, 3: 517, 4: 162}
Micro Average Precision Recall          F1      Over All Labels
                0.523       0.465       0.475

-------- Filtered Classifier --------
Read 156060 phrases, using 2500 random phrases
4849
Each fold size: 500

Average Precision       Recall          F1      Per Label
0               0.157       0.191       0.172
1               0.175       0.302       0.220
2               0.846       0.618       0.714
3               0.203       0.341       0.254
4               0.113       0.223       0.147

Macro Average Precision Recall          F1      Over All Labels
                0.299       0.335       0.301

Label Counts {0: 120, 1: 410, 2: 1294, 3: 530, 4: 146}
Micro Average Precision Recall          F1      Over All Labels
                0.523       0.464       0.476
```

S3 Stopwords Filter & Negations:

Explanation:

The stop-words and negations expressed earlier were filtered out of the classification evaluation to gain a simple understanding as to what a non-filtered corpus will produce (49%) verses a filtered (47%) corpus. Therefore, in this case a lower F1 score was obtained from the filtered version but the stop-words are not necessarily fostering the accuracy difference, rather is the negation of ‘don’t’ and other words of that nature. Moreover, it excludes tokens that perhaps determine a reviews sentiment and with a more though evaluation of the dataset unique/specific words will be negated that will encourage a better depiction of movie review sentiment.

Implemented Evaluation Results:

```
----- !STOPWORDS & NEGATION! -----

-------- Non_Filtered Classifier --------
Read 156060 phrases, using 2500 random phrases
4909
Each fold size: 500


Average Precision       Recall          F1      Per Label
0               0.136       0.170       0.136
1               0.233       0.355       0.280
2               0.827       0.608       0.701
3               0.262       0.421       0.322
4               0.128       0.306       0.178

Macro Average Precision Recall          F1      Over All Labels
                0.317       0.372       0.323

Label Counts {0: 98, 1: 449, 2: 1269, 3: 527, 4: 157}
Micro Average Precision Recall          F1      Over All Labels
                0.531       0.487       0.490

-------- Filtered Classifier --------
Read 156060 phrases, using 2500 random phrases
4753
Each fold size: 500

Average Precision       Recall          F1      Per Label
0               0.048       0.151       0.072
1               0.197       0.323       0.244
2               0.857       0.603       0.708
3               0.247       0.396       0.304
4               0.094       0.276       0.137

Macro Average Precision Recall          F1      Over All Labels
                0.289       0.350       0.293

Label Counts {0: 114, 1: 436, 2: 1275, 3: 534, 4: 141}
Micro Average Precision Recall          F1      Over All Labels
                0.532       0.471       0.479
```

Step 4 (NLTK vs. SciKit Learn Classifier)

S4 Scikit Learn Overview :

Scikit Learn is an open source Python learning library that started as a project for the 2007 Google Summer of Code. Since then its popularity has grown and it is currently known for being one of the most efficient tools building relevant machine learning statistical models. This tool NB Classifier is juxtaposed to NLTK's NB Classifier in order to understand which is more effective for building a classifier for Rotten Tomatoes Movie Reviews.

This NLTK Classifier is filtered by Stopwords, Negation and Non-Alphabetical characters.

S4 Filtered NLTK Accuracy Evaluations:

Explanation:

NLTK's NB Text Classifier unfolded a Macro Average F1 of 48% under the sample of 2,500/156,060. Although, the accuracy is lower than SK Learn it could prove to be an optimal classifier once a larger sample size of 80% (SK Learn sample size) instead of less than 10% training split. With fold 3 producing F1 of 70% classification against its tested counterpart, proving NLTK NB model is a better predictor for movie reviews.

Implemented Results:

```
 -------- NLTK Classifier BEFORE --------

Read 156060 phrases, using 2500 random phrases
4897
Each fold size: 500

Average Precision       Recall          F1      Per Label
0               0.101       0.146       0.116
1               0.222       0.337       0.266
2               0.841       0.599       0.699
3               0.245       0.433       0.312
4               0.060       0.156       0.084

Macro Average Precision Recall          F1      Over All Labels
                0.294       0.334       0.295

Label Counts {0: 109, 1: 405, 2: 1271, 3: 579, 4: 136}
Micro Average Precision Recall          F1      Over All Labels
                0.528       0.474       0.480

 -------- FULLY FILTERED NLTK Classifier --------

Read 156060 phrases, using 2500 random phrases
4747
Each fold size: 500

Average Precision       Recall          F1      Per Label
```

```
0             0.123      0.319      0.171
1             0.181      0.312      0.227
2             0.849      0.597      0.700
3             0.250      0.430      0.315
4             0.115      0.261      0.155

Macro Average Precision Recall          F1      Over All Labels
              0.304      0.384      0.314

Label Counts {0: 109, 1: 414, 2: 1289, 3: 558, 4: 130}
Micro Average Precision Recall          F1      Over All Labels
              0.535      0.483      0.485
```

S4 SK Learn Accuracy Evaluations:

Explanation:

The python code was sourced from a Kaggle user cited below however, the Multinomial Naive Bayes function utilizes the standard NB supervised learning algorithm that assumes conditional independence of each paired data-point. Which is useful for text classification especially with product or service reviews that have the review and sentiment as a paired association. Therefore, the code vectorized the 'PhraseID' and 'Sentiment' then gathers independent pairs probability in the dataset followed by smoothing its maximum likelihood. The results below convey a weighted F1 score of 53% and accuracy of 58% which is higher than NLTK's 48% indicating that perhaps Scikit Learn Classifier can predict with higher accuracy.

Implemented Results:

```
-------- SCIKIT LEARN Classifier --------

        PhraseId  SentenceId                                           Phrase  Sentiment
0              1           1  A series of escapades demonstrating the adage ...          1
1              2           1  A series of escapades demonstrating the adage ...          2
2              3           1                                         A series          2
3              4           1                                                A          2
4              5           1                                           series          2
...          ...         ...                                              ...        ...
156055    156056        8544                                       Hearst 's          2
156056    156057        8544                          forced avuncular chortles          1
156057    156058        8544                                 avuncular chortles          3
156058    156059        8544                                          avuncular          2
156059    156060        8544                                           chortles          2

[156060 rows x 4 columns]
A series of escapades demonstrating the adage that what is good for the goose is also good for the
```

```
gander , some of which occasionally amuses but none of which amounts to much of a story . Sentiment -
1
A series of escapades demonstrating the adage that what is good for the goose Sentiment -  2
for the gander , some of which occasionally amuses but none of which amounts to much of a story
Sentiment -  2


the gander , some of which occasionally amuses but none of which amounts to much of a story Sentiment -
1
              precision    recall  f1-score   support

           0       0.62      0.03      0.07      1443
           1       0.51      0.25      0.34      5386
           2       0.60      0.90      0.72     15876
           3       0.52      0.37      0.44      6629
           4       0.66      0.05      0.10      1878

    accuracy                           0.58     31212
   macro avg       0.58      0.32      0.33     31212
weighted avg       0.58      0.58      0.53     31212


  Cite_Author_Name_      =  https://www.kaggle.com/harshitmakkar

  Cite_Notebook_URL      =  https://www.kaggle.com/harshitmakkar/sentiment-analysis-on-movie-reviews-nlp

  Cite_SK_Learn NB Model =
https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
```

Code Sample:

```
def skLearn(dirPath,split_size):
  import pandas as pd
  import sklearn
  from sklearn.model_selection import train_test_split
  from sklearn.feature_extraction.text import CountVectorizer
  from sklearn.feature_extraction.text import TfidfTransformer
  from sklearn.naive_bayes import MultinomialNB
  from sklearn.pipeline import Pipeline
  from sklearn.metrics import classification_report

  # ------------------------------------------------------------ #
  os.chdir(dirPath)
  f = open('./train.tsv', 'r')
  data = pd.read_csv('./train.tsv',sep='\t')
  # ------------------------------------------------------------ #
  X = data['Phrase']
  y = data['Sentiment']
  # ------------------------------------------------------------ #
```

```
  phrase_train,phrase_test,sentiment_train,sentiment_test =
train_test_split(X,y,test_size=split_size)
  pipeline = Pipeline([('vect',CountVectorizer()),
                       ('tfidf',TfidfTransformer()),
                       ('classifier',MultinomialNB())])
  pipeline.fit(phrase_train,sentiment_train)
  predictions = pipeline.predict(phrase_test)
  print(classification_report(sentiment_test,predictions))
```

Conclusion (Analysis Overview)

Understanding the reviewers intentions or meaning is not easily understandable, especially when the objective is to predict how well a potential movie will perform amongst the public. Therefore, this paper captured and processed an open sourced movie review from Rotten Tomatoes and tested how the words from each review should be molded into a usable format that can be utilized in a movie review prediction models. Moreover, after understanding how different features impact each token, then a final product that is ready to classify movie reviews as negative, somewhat negative, neutral, somewhat positive, positive.

After the movie reviews patterns are understood mathematically (Naive Bayes) and stored in computer memory it is then tested against reviews without sentiment labels, in order to convey what would by these particular review sentiment. As a result, NLTK classify seems the most efficient and productive manner which can increase movie review sentiment understanding. The classifier is using cross validation to improve model however, a narrow approach of only one source of data can lead to overfitting which is handicapping to classifier to only understand a particular dataset and not being able to generalize its results.Therefore, addressing that newly released movies with reviews can be analyzed and presented to movie makers whether or not their investment is increasing

References

Cross Validation

Medium, Towards

Wikipedia

Kaggle Dataset

Rotten Tomatoes Dataset

SK Learn Classify Model Sampled

NLTK Naive Bayes Classifier

Classifying Text Ch.6

SciKit Learn NB Classifier:

SK Learn Multinomial

Cross Validation Methods

Sentiment analysis API and Documentation

Paralleldots / Sentiment-Analysis

Papers:

The Evolution of Sentiment Analysis - A Review of Research Topics, Venues, and Top Cited Papers

Precision Recall & F-Measure

F1 Measure Explanation

Precision and Recall Explanation