Natural Language Processing
(IST-664 2022-1005)

Assignment: Final Project

Due On: December 18, 2022

INSTRUCTIONS

- Final Project is a classification task.
- Develop features for the task.
- Show you can identify which feature sets are best for data & carry out experiments
- Task is designed to be appropriate for a single person
  -- May also work in groups of 2-3 people.
  -- For each additional person increase by 50%.

Dataset produced for the Kaggle competition

- Described here: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews
- Sentiment analysis by Socher et al, detailed at this web site: http://nlp.stanford.edu/sentiment/.
- Original Pang and Lee movie review corpus
  – Based on reviews from Rotten Tomatoes website.
- 5 categories: “neg”, “somewhat neg”, “neutral”, “somewhat pos”, “pos”.
- Use the training data “train.tsv”, and some test data is also available “test.tsv”.
- There are 156,060 phrases in the training data file.
  -- A challenge will be to select a training subset for processing & training.
- Each Sentence has been parsed into many phrases by the Stanford parser.
  -- Each phrase has a PhraseId and each sentence has a SentenceId.
  -- Phrases that are repeated (such as short/common words) are only included once in the data.

What to Submit: Each Person

- Report with the description of all that you did and the discussion of the results.
- Submit any python programs that you write, particularly to show additional features that you implement, or the annotation files that you did.
- If Group:
  -- The report should include which tasks each person worked on.
  -- Each person in the group should submit the (same) report.

* The final report will be due by the end of Week 11 (11:59pm EST, Sunday).

PART 0: Introduction

The focus of this project was text classification for sentiment analysis. The team looked at movie reviews using the Rotten Tomatoes dataset from Kaggle (https://www.kaggle.com/competitions/sentiment-analysis-on-movie-reviews/data).

The data has been preprocessed slightly. 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.

The train.tsv file contains the phrases and their associated sentiment labels. A SentenceId is also provided so that phrases which belong to a single sentence can be tracked. The test.tsv file contains just phrases. The sentiment labels are:

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

The object of this project is to create and compare models in order to find the ones that provide the best accuracy, precision and recall for the classification of the dataset.

PART 1: Data Understanding and Preparation

The data were downloaded from the Final Project folder provided to the class by the professor. The folders were unzipped and copied to a shared Google folder for the team. A Google Colab file was created for ease of sharing code.

The files were read into a Pandas dataframe and some basic information was gathered about the data.

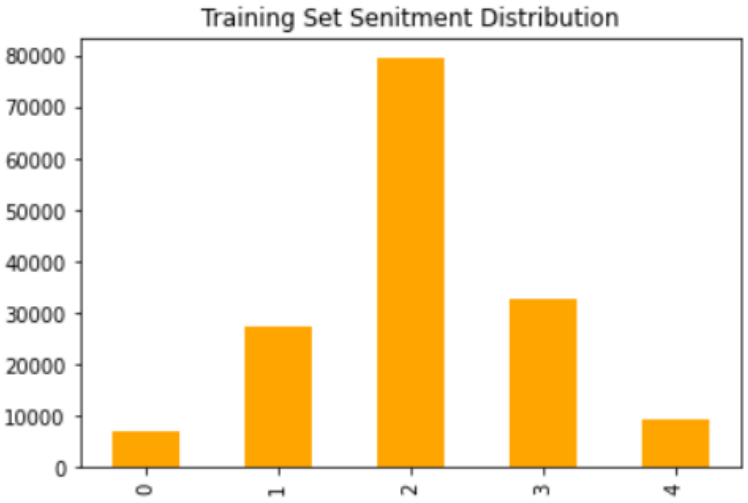
The training data file contains 156,000 entries. Each entry has a PhraseId, SentenceId, Phrase (the text of the review), and Sentiment rating.

The test data file contains 66,292 entries. Each entry has the same data points as the training data file with the exception of a sentiment rating.

Number of Observations in the Data
Test Count of Observations: 66292.0 Train Count of Observation: 156060.0 Total Number of Observations(Test+Train): 222352.0 Test/Train Percentage Split: 29.8 / 70.2

Training Data	Test Data
<class 'pandas.core.frame.DataFrame'> RangeIndex: 156060 entries, 0 to 156059 Data columns (total 4 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 PhraseId 156060 non-null int64 1 SentenceId 156060 non-null int64 2 Phrase 156060 non-null object 3 Sentiment 156060 non-null int64 dtypes: int64(3), object(1) memory usage: 4.8+ MB	<class 'pandas.core.frame.DataFrame'> RangeIndex: 66292 entries, 0 to 66291 Data columns (total 3 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 PhraseId 66292 non-null int64 1 SentenceId 66292 non-null int64 2 Phrase 66292 non-null object dtypes: int64(2), object(1) memory usage: 1.5+ MB

A look at the frequency distribution of the ratings shows that the data follows the pattern of a normal distribution with the most frequent ratings being a 2.



Sentiment Frequency in Training Set:

0 7072
1 27273
2 79582
3 32927
4 9206
Name: Sentiment, dtype: int64

Next a sample of the phrases from the different ratings classes was explored.

Negative Phrases Samples

	PhraseId	SentenceId	Phrase	Sentiment
101	102	3	would have a hard time sitting through this one	0
103	104	3	have a hard time sitting through this one	0
157	158	5	Aggressive self-glorification and a manipulati...	0
159	160	5	self-glorification and a manipulative whitewash	0
201	202	7	Trouble Every Day is a plodding mess .	0
...	...	...	...	...
155965	155966	8539	has turned out nearly 21\/2 hours of unfocused...	0
155967	155968	8539	turned out nearly 21\/2 hours of unfocused , e...	0
155970	155971	8539	of unfocused , excruciatingly tedious cinema	0
155971	155972	8539	unfocused , excruciatingly tedious cinema	0
155973	155974	8539	, excruciatingly tedious	0

7072 rows × 4 columns

Somewhat Negative Reviews Samples

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
33	34	1	the gander , some of which occasionally amuses...	1
47	48	1	but none of which amounts to much of a story	1
49	50	1	none of which amounts to much of a story	1
81	82	3	Even fans of Ismail Merchant 's work , I suspe...	1
...	...	...	...	...
156036	156037	8543	substitute plot for personality	1
156047	156048	8544	quietly suggesting the sadness and obsession b...	1
156051	156052	8544	sadness and obsession	1
156052	156053	8544	sadness and	1
156056	156057	8544	forced avuncular chortles	1

27273 rows × 4 columns

Neutral Phrases Samples

	PhraseId	SentenceId	Phrase	Sentiment
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
5	6	1	of escapades demonstrating the adage that what...	2
...	...	...	...	...
156053	156054	8544	beneath Hearst 's forced avuncular chortles	2
156054	156055	8544	Hearst 's forced avuncular chortles	2
156055	156056	8544	Hearst 's	2
156058	156059	8544	avuncular	2
156059	156060	8544	chortles	2

79582 rows × 4 columns

Somewhat Positive Samples

	PhraseId	SentenceId	Phrase	Sentiment
21	22	1	good for the goose	3
22	23	1	good	3
46	47	1	amuses	3
64	65	2	This quiet , introspective and entertaining in...	3
67	68	2	quiet , introspective and entertaining	3
...	...	...	...	...
156012	156013	8541	great job	3
156014	156015	8541	anchoring the characters in the emotional real...	3
156023	156024	8542	-LRB- Tries -RRB-	3
156043	156044	8544	is darkly atmospheric	3
156057	156058	8544	avuncular chortles	3

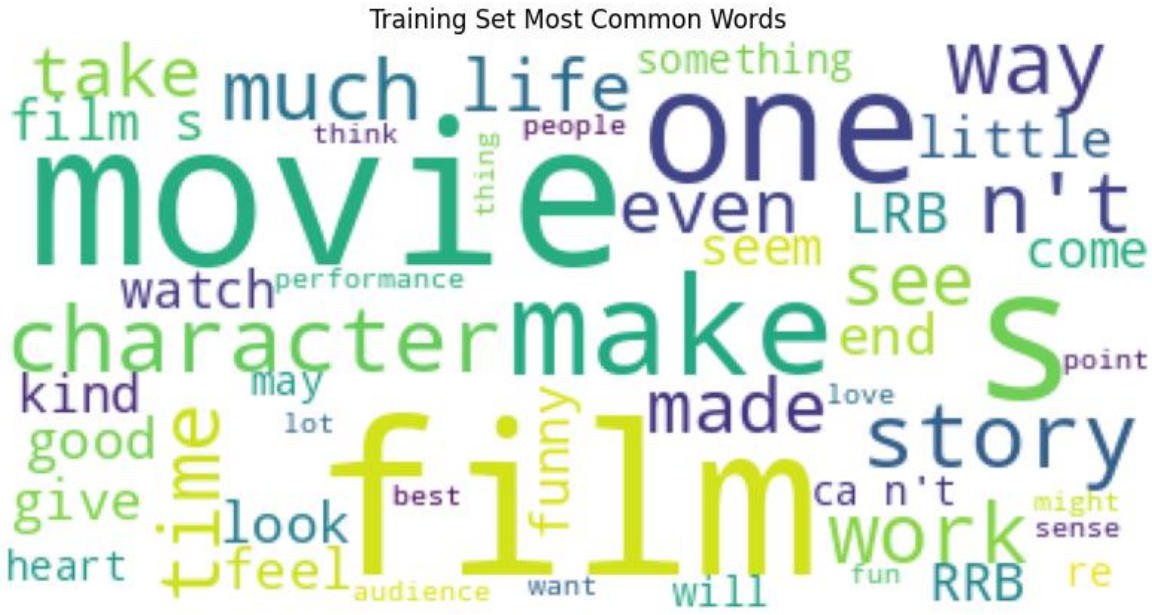
32927 rows × 4 columns

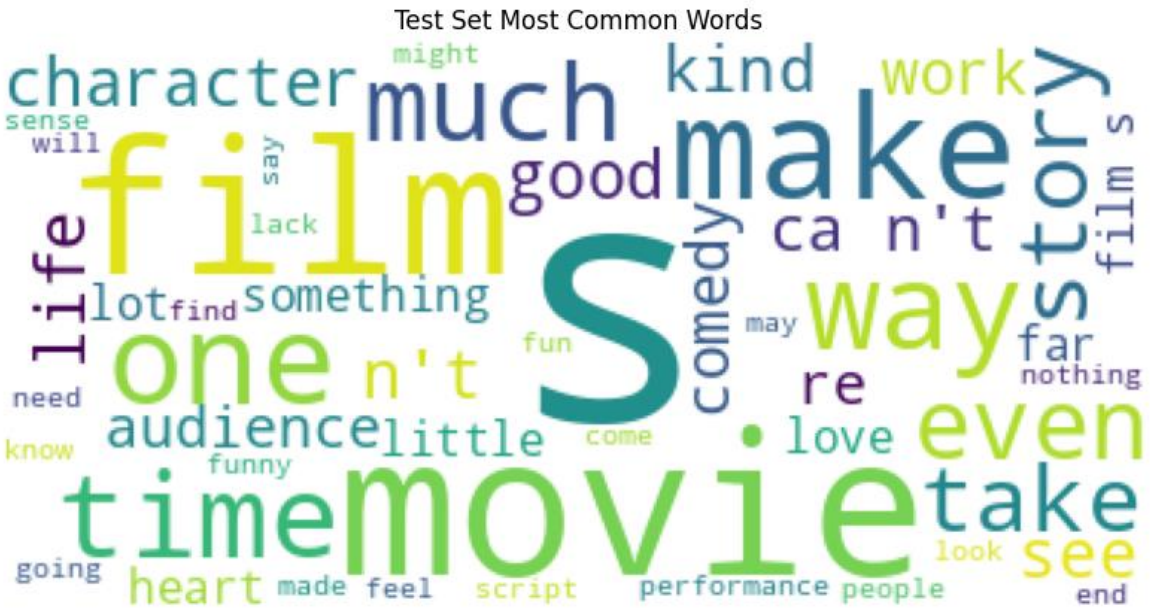
Positive Samples

	PhraseId	SentenceId	Phrase	Sentiment
63	64	2	This quiet , introspective and entertaining in...	4
66	67	2	quiet , introspective and entertaining indepen...	4
74	75	2	entertaining	4
77	78	2	is worth seeking	4
117	118	4	A positively thrilling combination of ethnogra...	4
...	...	...	...	...
155946	155947	8537	is laughingly enjoyable	4
155955	155956	8538	a unique culture that is presented with univer...	4
155961	155962	8538	with universal appeal	4
156007	156008	8541	really do a great job of anchoring the charact...	4
156010	156011	8541	a great job of anchoring the characters in the...	4

9206 rows × 4 columns

Last step was to create some Word Clouds to provide a visual representation of the word frequencies in the two data sets. As can bee seen below, there is much overlap between the two sets which is good for the analysis.





Data Preprocessing

The full training data was extremely large. As a result, the computational power necessary to use all of the training data was beyond our computational means. This required the team to work with a subset of the original training data. A function was written to ensure that the selected subset of data was both random and still maintained the category proportions. In this way, we could vary the subset size while still ensuring it was a representative sample of the training data.

The code itself was written in a modular design using functions. This reduced the program complexity and ensured consistency between the experiments. While Google Colab was the software development platform of choice it quickly became apparent that personal computers would be required as we easily exceeded the 12 Gigabyte RAM Limitation.

Due to these limitations and after some testing with a variety of sizes, 5% of the original training data set was selected as the size to use for the NLTK and other experiments. In order to ensure the selected data maintained the category relative frequency proportion the data was split using the ‘train_test_split’ function from the ‘sklearn.model_selection’

Part 2 - Feature Functions in Python

Baseline Feature Production

Before creating any feature sets, the team wanted to get a baseline number using the NLTK Naive Bayes classifier for the purpose of comparison with different feature sets. Because we wanted a baseline, very minimal preprocessing was done to the data. The standard steps of tokenization, lower casing and removing punctuation performed. Stopwords were not removed for this round of classification. A vocabulary size of 200 words was set.

The classification was run 4 times, varying the size selected from the training data set from 5% to 20%. The results were then run through a cross validation. Above 20%, the program ran into the limitations of the computing resources available and was not able to successfully complete. The team decided that a baseline of 5% of the data and a 90/10 training/testing split would be used.

EXPERIMENT 0: Establish Baseline				
Vocabulary Size	1000			
Training/Test Split	90/10			
% of Data	Precision	Recall	F1	Accuracy
0.05	0.344154	0.378546	0.348668	0.537308
0.10	0.355965	0.389796	0.35946	0.539955
0.15	0.364963	0.397628	0.367924	0.542619
0.20	0.371976	0.397628	0.374071	0.543448
Test 2				
Vocabulary Size	200			
Training/Test Split	90/10			
% of Data	Precision	Recall	F1	Accuracy
0.05	0.312601	0.340875	0.31088	0.517308
EXPERIMENT 0: RECOMMENDATIONS: Due to time and computer resources.				
Total Data to Use	5%			

Training/Test Split	90/10			

The next experiment was designed to define the optimal vocabulary size for the training set as defined in Experiment 0. The test was run 5 times using the Naive Bayes classifier while varying the number of words used for the vocabulary between 100 and 3000. The results were then run through a cross validation. The results showed that a vocabulary size of 1000 would produce an accuracy of around 54% while still allowing the code to execute.

EXPERIMENT 1: Determine Optimal Vocabulary Size				
Training Data	15%			
Training/Test Split	80/20			
Vocabulary Size	Precision	Recall	F1	Accuracy
100	0.30765	0.337783	0.304545	0.511386
500	0.350946	0.379406	0.350505	0.529289
1000	0.36614	0.396794	0.368897	0.540782
1500	0.374615	0.40389	0.3777	0.546336
3000	0.389056	0.418762	0.393862	0.555266
Test 2				
Training Data	5%			
Training/Test Split	90/10			
Vocabulary Size	Precision	Recall	F1	Accuracy
200	0.312601	0.340875	0.31088	0.5230769231
500	0.325922	0.353100	0.325152	0.5423076923
1000	0.345486	0.379523	0.350139	0.5512820513
1500	0.346728	0.383592	0.352496	0.5487179487
3000	0.341644	0.388807	0.352537	0.5371794872
Test 3				
Training Data	10%			

Training/Test Split	90/10			
Vocabulary Size	Precision	Recall	F1	Accuracy
1500	0.369407	0.402361	0.37323	0.5554131967
EXPERIMENT 1: RECOMMENDATIONS: Due to time and computer resources.				
Total Data to Use	5%			
Training/Test Split	90/10			
Vocabulary Size	1500			

Part 3 - Experiments

Experiment 2: Stopwords

To further refine the parameters for the feature sets, the next step the team took was to remove stopwords from the vocabulary. The features were generated with no stop words, the basic stopwords list from NLTK and then a custom set of stopword lists made up of the basic list plus a number of most common and least common words. The final list of stopwords selected was the Basic NLTK set of 179 English words plus a custom set of 15 words: [',', '.', "'s", 'film', 'movie', "n't", '--', '"', '-rrb-', '-lrb-', '`', "'", '``', '...', ':'].

EXPERIMENT 2: Explore Stop Words				
Training Data	5%			
Training/Test Split	90/10			
Stopwords	TRUE (Basic NLTK Set + Extended)			
Stopwords Removed	Precision	Recall	F1	Accuracy
Basic + 0	0.341032	0.394688	0.353892	0.5641025641
Basic + 15	0.33682	0.400832	0.350878	0.5628205128
Basic + 22	0.327988	0.389928	0.340755	0.5256410256
Basic + 30	0.328711	0.392086	0.34264	0.5564102564

EXPERIMENT 2: RECOMMENDATIONS: Due to time and computer resources.				
Total Data to Use	5%			
Training/Test Split	90/10			
Vocabulary Size	1500			
Stopwords	Basic + 15			

Experiment 3: Bigrams

The third experiment looked at the effect of bigrams on the accuracy of the Naive Bayes classifier. Here we used chi_sq to find the best frequency relationship in bigrams of various sizes (amount of bigrams). Then we created feature sets from those bigrams and trained our naive bayes classifier using our same 90/10 split. Our accuracy improved the larger the size of bigrams used. So, a recommendation of 5% with a size of 3,000 bigrams and a 90/10 train/test split seems optimal. No stop words were used for these results.

EXPERIMENT 3: Bigrams Investigation				
Training Data	5%			
Training/Test Split	90/10			
Bigrams Size	Precision	Recall	F1	Accuracy
200	0.297981	0.373607	0.307671	0.5269230769
500	0.302298	0.38817	0.313569	0.5256410256
1000	0.299846	0.385136	0.311004	0.5397435897
1200	0.35276	0.390136	0.358187	0.5551282051
3000	0.340605	0.375453	0.345114	0.5833333333

EXPERIMENT 3: RECOMMENDATIONS: Due to time and computer resources.				
Total Data to Use	5%			
Training/Test Split	90/10			
Bigrams Size	3000			

Sentiment Lexicon

A feature set was created using the subjectivity lexicon file from Wiebe et al at http://www.cs.pitt.edu/mpqa/ (part of the Multiple Perspective QA project). The training set was used to create a new tagged feature set with each phrase being given one of 4 sentiment classes, weak positive, strong positive, weak negative and strong negative. These tags were added to the feature set and then used with the Naive Bayes Classifier and passed through the cross validation function.

As can be seen below, the sentiment lexicon provides about the same accuracy as the other models, coming in at 56%.

EXPERIMENT 4: Sentiment Lexicon				
Training Data	5%			
Training/Test Split	90/10			
Vocabulary Size				
Lexicon	Precision	Recall	F1	Accuracy
Subjectivity Lexicon	0.379473	0.400017	0.386759	0.556026
EXPERIMENT 4: RECOMMENDATIONS: Due to time and computer resources.				
Total Data to Use	5%			
Training/Test Split	90/10			
Lexicon	yes			

Sklearn Models

In order to try and find a better model with greater accuracy, a few other classifiers from the Scikit-Learn library were used. The models used were the LinearSVC, Multinomial Naive Bayes and several flavors of Logistic Regression. The feature set used with these models was the original set of features without stop words removed. This set of 1561 features were changed to lowercase and then run through a program to create a new frequency distribution and write them all to a csv file in matrix format. This new sklearn feature set was then used with all 3 models. In addition, several other test runs were performed with a variety of vocabulary sizes from 100 to 2000.

These models all generated an accuracy score as well as performing cross validation. As can be seen in the table below, all of the models performed about the same. Accuracy hovered around 50-54% across all vocabulary sizes and models.

EXPERIMENT 5: SKLEARN Models				
Training Data	5%			
Training/Test Split	90/10			
Vocabulary Size 1000	Precision	Recall	F1	Accuracy
Linear Regression - newton-cg	0.36	0.38	0.37	0.50
Linear Regression -lbfg	0.35	0.39	0.36	0.48
Linear Regression - liblinear	0.38	0.38	0.37	0.53
Vocabulary Size 1500	Precision	Recall	F1	Accuracy
Linear Regression - newton-cg	0.37	0.38	0.37	0.51
Linear Regression -lbfg	0.36	0.39	0.37	0.50
Linear Regression - liblinear	0.39	0.38	0.38	0.54
Linear SVM	0.30	0.26	0.3	0.50
Vocabulary Size 2000	Precision	Recall	F1	Accuracy
Linear Regression - newton-cg	0.38	0.38	0.38	0.52

Linear Regression -lbfg	0.37	0.39	0.38	0.50
Linear Regression - liblinear	0.40	0.38	0.38	0.54
EXPERIMENT 5: RECOMMENDATIONS: Due to time and computer resources.				
Total Data to Use	5%			
Training/Test Split	90/10			
SKLearn Model	none			

In addition, the models generated a predicted vs actual matrix to show how well the model did at correctly predicting the labels. Across the board, the models were very good at correctly predicting reviews with a score of '2'. For all others, the models had similar poor performance on predicted vs. actual. Overall, these models did not add any enhanced performance to the classification accuracy.

Logistic Regression - newton-cg

Predicted	0	1	2	3	4	All
Actual						
0	83	112	89	26	24	334
1	154	359	607	183	48	1351
2	129	474	2797	498	99	3997
3	52	180	679	580	192	1683
4	23	37	74	157	147	438
All	441	1162	4246	1444	510	7803

Logistic Regression - lbfgs

Predicted	0	1	2	3	4	All
Actual						
0	97	101	76	33	27	334
1	166	385	556	189	55	1351
2	167	515	2655	535	125	3997
3	61	194	616	575	237	1683
4	29	40	64	143	162	438
All	520	1235	3967	1475	606	7803

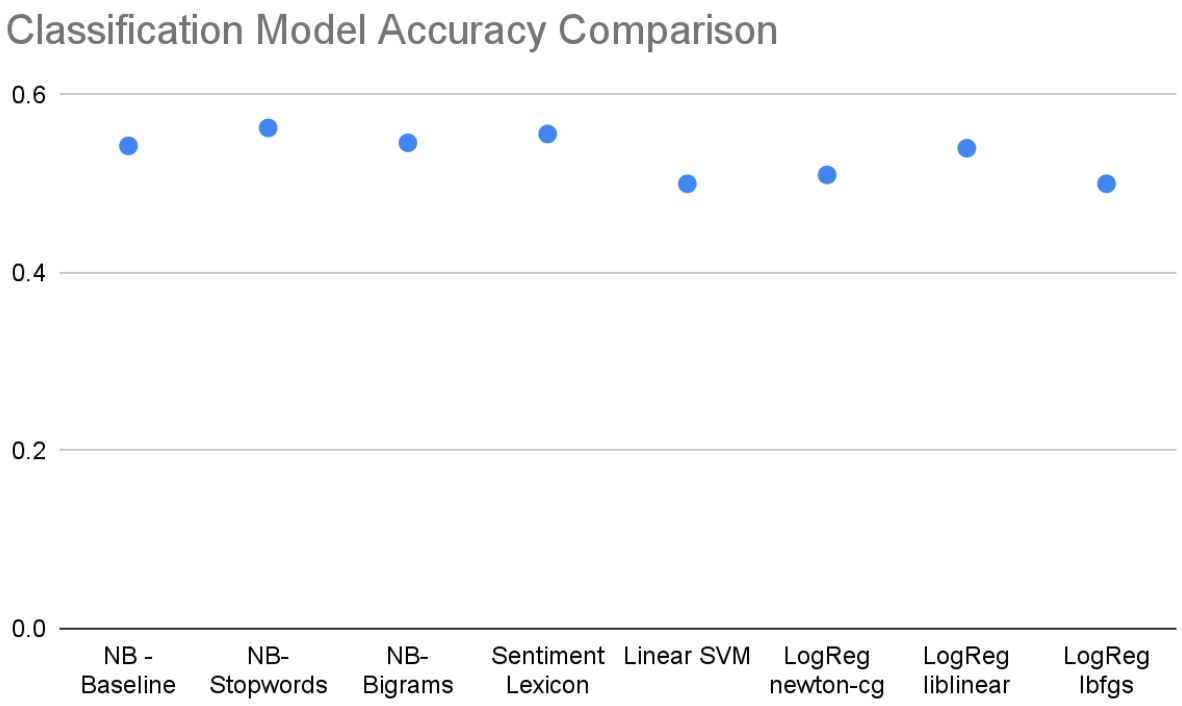
Logistic Regression - liblinear

Predicted	0	1	2	3	4	All
Actual						
0	81	100	107	26	20	334
1	132	317	721	138	43	1351
2	112	304	3155	344	82	3997
3	45	148	819	492	179	1683
4	23	32	105	135	143	438
All	393	901	4907	1135	467	7803

PART 4: Results and Discussion

For this classification task a variety of features sets and models were used to try to ascertain the best set of variables to use to correctly predict ratings scores for the Kaggle movie reviews data set. The table below shows a comparison of the outputs from the different models. The data that produced the best results given the constraints of computing resources was a 5% random sample of the full data set that was then split into a 90% training set and a 10% testing set using a vocabulary of 1500 most common words.

Model Comparison					
Training Data	5%				
Training/Test Split	90/10				
Vocabulary Size	1500				
Model		Precision	Recall	F1	Accuracy
NB - Baseline		0.364963	0.397628	0.367924	0.542619
NB- Stopwords		0.336820	0.400832	0.350878	0.562821
NB- Bigrams		0.359529	0.391573	0.364608	0.546095
Sentiment Lexicon		0.379473	0.400017	0.386759	0.556026
Linear SVM	macro average	0.30	0.26	0.30	0.50
	weighted average	0.42	0.50	0.42	
LogReg newton-cg	macro average	0.37	0.38	0.37	0.51
	weighted average	0.50	0.51	0.50	
LogReg liblinear	macro average	0.39	0.38	0.38	0.54
	weighted average	0.51	0.54	0.51	
LogReg lbfgs	macro average	0.36	0.39	0.37	0.50
	weighted average	0.50	0.50	0.50	



The results of the models show that accuracy for all hovered around 55% with the Naive Bayes using a custom set of stop words and the Sentiment Lexicon performing the best with 56% accuracy each. These results show that there is possibly more work to be done in tuning the stop words list to get better performance from these models.

PART 5: Conclusion

This dataset proved to be challenging to correctly categorize. Multiple features and classification algorithms were used in an attempt to properly classify the movie reviews. In all, the best method only produced an accuracy of 56%. This was a bit disappointing but given the fact that this dataset was used in a Kaggle Competition perhaps it is not surprising how tough this task was.

While much was accomplished in this project there are many more opportunities to further explore this data. If time had permitted additional customized feature sets could have been explored as well as other types of machine learning algorithms.