Problem Setup

In this Experiment, I first started with the preprocessing stage in order to clean all the data extracted from rt-polarity. I converted all characters to lowercase, and removed any special characters. After which I used NLTK's library to import the set of stop words and remove them from my data. Moreover, I created preprocessing methods which could be applied to my models if chosen to stem or lemmatize the tokens. Finally, I used Scikit-learn's CountVectorizer to transform the corpora to a numerical vector representation of words and n-grams. I also made use of the 'train_test_split' from Scikit-learn's library which allowed me to easily split the dataset into the training and the testing datasets in an 80%-20% proportion which was a constant in this experiment.

Experimental Procedure

The training data was then used to train four different models: Naive Bayes, Logistic Regression, Support Vector Machine and Stochastic Gradient Descent (SGM). The SGM Classifier was chosen in order to investigate whether it would improve the accuracy of the linear classifiers due to its optimization properties. Moreover, a random baseline which just guesses positive or negative with equal probability was used to compare against. The test data was then used to obtain predicted results from each model. I was then able to measure the model's accuracy using 'accuracy_score' from Scikit-learn's metrics library, which compared the predicted results against the set of true data.

Parameter Settings

In order to investigate how different models would perform with varying inputs, I recorded the accuracy results when the dataset was: only stemmed, only lemmatized. These varying parameters were tested for varying min_df values of 1, 2 and 3 which is used for removing terms that appear too infrequently.

Results

The accuracy values tabulated below are the average of 5 runs.

Classifier	**Min_df = 1**		**Min_df = 2**		**Min_df = 3**	
	Lemmatized	Stemmed	Lemmatized	Stemmed	Lemmatized	Stemmed
Logistic Regression	0.756	0.744	0.751	0.741	0.751	0.743
Linear SVM	0.732	0.724	0.719	0.707	0.708	0.707
Stochastic Gradient Descent	0.735	0.724	0.723	0.719	0.721	0.722
Naive Bayes	0.768	0.759	0.771	0.763	0.769	0.765
Random Baseline	0.518	0.511	0.500	0.514	0.495	0.497

Confusion Matrix: Naive Bayes, min_df = 2, Lemmatized

n= 2133	**Positives**	**Negatives**
Positives	True Positives: 852	False Positives: 259
Negatives	False Negatives: 229	True Negatives: 793

Precision: 852/852+259 = 0.767
Accuracy: (852+793)/2133 = 0.771

Conclusion

We can see from the results that there was an overall better performance when performing lemmatization on the words. This would make sense, given that the stemmer operates on a single word without knowledge of context and is therefore not able to discriminate between words which have different meanings. In lemmatization however, it will return the dictionary form of a word, which tends to convert the word to its meaningful base form by identifying the Part of Speech. The Naive Bayes Classifier had the best performance out of all the models in all cases, which also makes sense given its advantage in being able to classify large numbers of words as independent features and doesn't need much data to perform well. The SGM Classifier It was expected that the random baseline classifier would have an accuracy of ~50% and it acted as a good reference point to compare the accuracy of the rest of our models against. There wasn't a significant difference in accuracy results when min_df was changed.