Assignment 2: Sentiment Mining:

Task: The Assignment consisted of creating a Sentiment Classifier that will predict a given tweet to have a positive or negative sentiment.

System Description:

Text was first converted into vector representations, i.e. numericalizing texts so that SVM (and most other machine learning classifiers) can be applied. First I started with Multinomial Naïve Bayes considering only unigrams features. By Training on 80% of tweet data and testing on rest 20% I got 77% of accuracy. But there were several limitations in this because of the level of data sparsity, the size of feature space and performance. [1]
Data Sparsity Reduction: Since the size of bigram features was of order (100K) in size, it becomes crucial to reduce feature size, so as to ensure faster performance and reduce irrelevant words such as 'the', 'and', 'a', etc. I used Tf-Idf method based weighting of SkLearn Library, gave list of common stop words, used minimum frequency threshold. Employing these I was able to reduce Vocabulary size by 5.
Feature Selection: After applying filter as described above I used Bigrams Features set on Bag of Words representation. 1st Tokenizing, then counting frequencies and also normalizing and weighting based on importance of tokens. This is achieved by CountVectorizer of Sklearn library [3]. On features I have also tried POS tagging by appending tags as prefixes to the words (but it proved to reduce overall accuracy). I also used Lemmatization as it helped me achieve 0.3 increase in accuracy over total Data. I also treated negatives by appending "NOT" to the next word. This also helped me increase accuracy by about 0.5%. For POS_Tagging, Lemmatizing I used NLTK Library [2]
Classifier Selection & Training: I tried training on several classifiers including Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Perceptron, Logistic Regression (also tried using L1&L2 regularization), SVM. Out of all these roughly every except SVM (with linear kernel function) were able to train on whole Data set of 1.6 million tweets in at most 10 minutes. Whereas SVM taking more than 7 hrs for 160,000 tweets still producing low accuracy of about 78%.
Testing: Of all the above classifiers Logistic Regression (without any penalty) proved to be scalable one (faster) and the performance wise high as at the end of the day I was able to get accuracy of about 82.1% (from 77% from where I started). (Follow the figure where graph is plotted against Data Size and Performance Accuracy). Hence I have used Logistic Regression Classifier for training above mentioned Features.

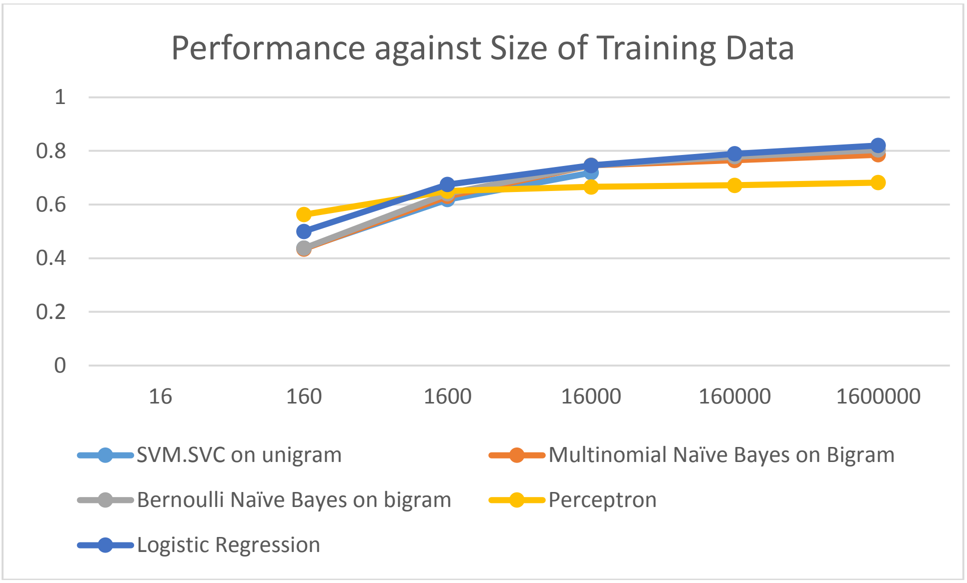

Fig.1 All these Classifiers trained for Bigram Features with POS Tagging.

Research and Observations:

Aim1: Filtering and Data Dimensionality Reduction on Sentiment Analysis Features.
Since the twitter data is very sparse and it becomes necessary to reduce dimension of feature as otherwise learning algorithms takes enormous amount of time in training data. Several methods were tried to achieve this. The vocabulary size was initially of order (100K). 1st I tried giving list of the common and irrelevant words in tweets and hence ignoring these from the twitter file. But this proved to be slower computationally and also gave negative performance accuracy. Then I tried Tfidf based weighing. This was faster to implement and also increased my accuracy 0.5%. I also gave giving minimum frequency threshold of 2. This reduced vocabulary size by scale of 5. Then I also tried KBestSelections, SelectPercentile methods of Sklearn Library to reduce dimensionality but these methods also reduced performance accuracy. Hence it can be inferred that giving pre-compiled stoplist has negative impact on accuracy. Hence, this can be seen as trade-off between Feature Size and Accuracy/ Performance. Hence as for Twitter Sentiment analysis removing singleton words achieves this trade-off between good performance and low computational processing time.

Aim2: Impact of performance on the size of Training Data for different classifiers.
From the above figure plotted it becomes clear that the performance of different classifiers tend to saturate as the size of data increases. Further the performance of Bernoulli Naïve Bayes (max about 81%), was comparable to that of Logistic Regression (max about 82%). The reason can be as Naïve Bayes classifiers are highly scalable, requires number of parameter linear in number of features. (In this case Training Data order = 1,200K and Feature Size of 100K). Also it was observed that Bernoullli Naïve Bayes performed better than Multinomial Naïve Bayes. Whereas SVM (with linear kernel function) was very slow for training data size of order = 16K. Hence after statistically observing performance of each classifier I chose Logistic Regression (without any regularisation penalty).

Aim3: Impact of different kinds of features:
I tried bigram features after trying unigram features, there was significant amount of performance by about 3-4% but also the feature size increased by about 9 times and of order = 100K size. Further I also tried adding POS tags to the words (however this proved to reduce accuracy), I also tried lemmatizing words, also appending NOT_ wherever I see a negative clitics or 'not'. With later two I was able to improve performance by about 0.5%. I also tried Trigrams on Naïve Bayes, initially it gave better performance on smaller training data but later saturated at around 79% compared to Bigrams saturating on 80%.

Refrences:
[1]: On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter
[2]: NlTK Library for Python
[3]: Scikit Learn Open Source Library for Python

Dhruvin Patel
2012CS50293