Opinion Mining of Amazon Text Reviews using Machine Learning Techniques

Abhiram Chepur -A20388946
Chanukya Srinivas Santhosh Kumar Patnam -A20384236

Introduction:

Classification is a technique in Statistics and Machine Learning which is used to identify to which category the new observation belongs to. In our project, we try to classify text reviews into positive, negative or neutral categories. This process of identifying the polarity of text is called Opinion Mining or Sentimental analysis. This process can be achieved through various Classification and Natural Language process techniques. The purpose of this project is to investigate a small part of this large problem: positive and negative attitudes towards products. Opinion Mining attempts to determine which features of text are indicative of it's context and build systems to take advantage of these features. The problem of classifying text as positive or negative is not the whole problem in and of itself, but it offers a simple enough premise to build upon further.

Previous Work:

Work involved in sentimental analysis in content containing personal opinions has been going on recently. Pang and Lee used several machine learning systems to classify a large carpus of movie reviews. Yessenov and Misailovi did similar work, taking comments off from social networking sites. Amazon employs a 1-to-5 scale rating for all products, and it becomes challenging to determine the advantages and disadvantages to different parts of a product.

Problems with such rating systems are discussed by Hu and Liu. Sentiment classification of Reviews by Sylvester Olubolu Orimaye is another such similar work which employed the usage of Sentence Polarity Shift algorithm.

Methodology:

Data:

Amazon does not have an API like twitter to download reviews, so we downloaded the data from UCSD Amazon product data by Julian McAuley. Digital Music reviews(reviews_Digital_Music_5.json.gz) with minimum 5 reviews for a product in JSON format was downloaded and converted in data frame format after parsing the data. Then the data frame has been trimmed to another data frame with only the useful columns i.e. "reviewerID","reviewText","overall", where overall is the overall rating of the product. Then the 'overall' has been normalized and a rating of 1(positive) was given for ratings greater than 3 and -1(negative) for others. Later we used another class i.e. 0 (neutral) for a rating of 3. The data has been divided into training data and test data. Train data has 44194 rows and test data has 19412 rows. Then the data has been cleaned (preprocessing) by removing all the HTML tags, non-letters and then splitting the data to individual words and removing the stop words. Then the words have been rejoined to make meaningful and clean reviews.

Feature Extraction:

Feature extraction has been done using the Bag of Words method. A bag of words feature vector consists of all the words in the article as independent features. In this method, all the words are collected from reviews and the subjective score of each word is obtained. The bag of words feature model assumes the independence of each of the words, it ignores some of the relationships between words that add affect their meanings in the context of the article i.e. features as unigrams were used. This method has been implemented using the Count Vectorizer function in the sklearn library. After this step, we get all the features used to train our models.

Baseline Performance:

Distinguishing positive from negative reviews is relatively easily for humans, especially when the whole review is read properly. But even a random guess has a probability of 50% to be correct. For three classes, we have a probability of 33.33%. We can also see that there are certain words that people use to express sentiments. This word list might suffice to depend on them to classify text. A test performed on a human generated keyword explained in the paper 'Thumbs up? Sentiment Classification using Machine Learning Techniques' gives accuracy of 58%. These provide us with baselines for experimental comparisons with our results.

Modelling Techniques:

Random Forests: Random forests algorithm is a supervised classification algorithm, it creates the forest with many trees. Generally, the more the trees the more robust the forest looks like. Therefore, in the random forest classifier the higher the number of trees the higher accuracy results. In random forest, we build many decision trees on bootstrapped training samples, when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from full set of p predictors. At the split only one of those m predictors is allowed, a new set of m predictors is chosen at each split, generally we take $m \approx \sqrt{p}$ i.e., the number of predictors considered at each split is approximately equal to square root of the total number of predictors. The feature vectors are used to train the random forest classifier in sklearn library and then test set is used to predict the polarity of the reviews with the trained Random Forests model.

Naïve Bayes: Naïve Bayes is a classifier applied using Bayes Rules. It assumes that each feature is independent and calculates the probability the feature appears in the given class. The probability of a class given the feature set is the product of the probability that the class will occur and the probability of each of the feature vectors. The process is repeated for all the classes and text is classified according to the maximum probability. More formally,

$c = \text{argmax } c \in C \; P(c \mid d) = \text{argmax } c \in C \; P(d \mid c) \; P(c) \; P(d) = \text{argmax } c \in C \; P(d \mid c) P(c)$

where c is the most likely class. First the feature vectors from the training set are used to train the Multinomial Naïve Bayes classifier in sklearn library and then test set is used to predict the polarity of the reviews with the trained Naïve Bayes model.

Support Vector Machines: Support Vector Machines are large-margin, rather than probabilistic classifiers. The idea is to find a hyperplane that separates the document vectors in one class from the other. SVM's maximize the margin around separating hyperplane. First the feature vectors from the training set are used to train the SVM classifier in sklearn library and then test set is used to predict the polarity of the reviews with the trained SVM model.

Logistic Regression: Logistic regression is a linear model, it assumes a linear relationship between the predictors and log odds of a classification. This uses the maximum likelihood method to fit the model. More formally, $P(x) = e^{B0+B1(x)}/1+ e^{B0+B1(x).}$ The coefficients of B0 and B1 are estimated from the training data and these are used to predict classes in the test data set. The feature vectors from the training set are used to train the Multinomial Logistic Regression classifier in sklearn library and then test set is used to predict the polarity of the reviews with the trained Logistic Regression model.

Performance Metric:

Accuracy is used as a performance metric. This gives us the performance of the algorithms in the sense how many of the predictions made are correct. Accuracy is the basic measure to test the performance of the algorithm. The accuracy that should satisfy depends on the reliability that the customer would have on it and the costs matrix of false predictions.

Results:

The classification accuracies resulting from using different algorithms are shown in the table below. The weighted F-score i.e., average, weighted by support is also calculated.

For two class labels:

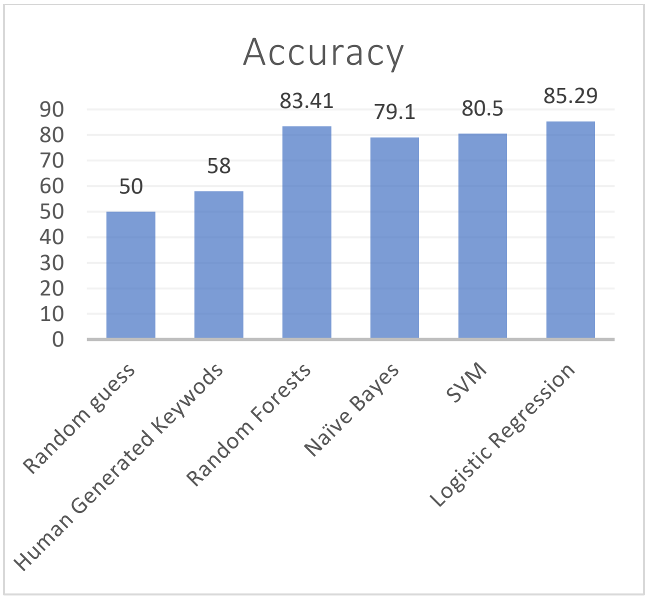
Algorithm	Features	Accuracy	F-score(weighted)
Random Forests	Unigrams	83.41%	0.7889
Naïve Bayes	Unigrams	79.1%	0.8026
SVM	Unigrams	80.50%	0.7206
Logistic Regression	Unigrams	85.29%	0.8479

For Three class labels:

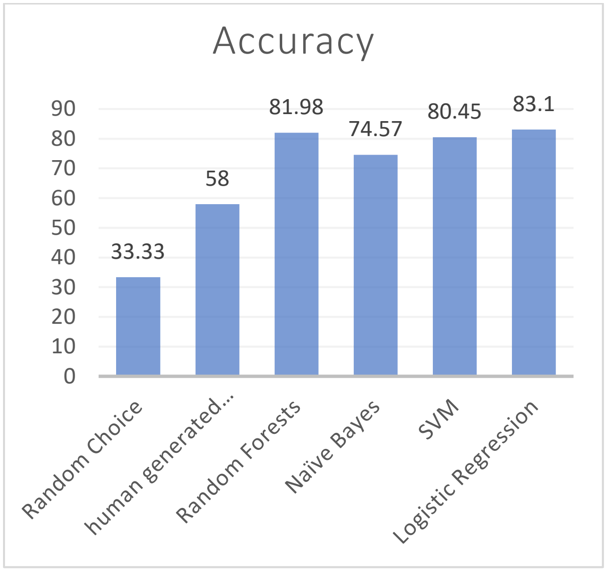
Algorithm	Features	Accuracy	F-score(weighted)
Random Forests	Unigrams	81.98%	0.7528
Naïve Bayes	Unigrams	74.57%	0.7642
SVM	Unigrams	80.45%	0.7174
Logistic Regression	Unigrams	83.10%	0.8169

From the above table, we can see that all our algorithms have comfortably surpassed the accuracy random guess baseline of 50%. Also, all the algorithms have performed better than the human generated keyword baseline of 58%. Now coming to the comparison between algorithms. We can see that Logistic Regression outperformed all the other classifiers in the case of both when using two class labels and three class labels. Even though our intuition tells us that Logistic regression performs better only with two class labels, we can see that in this case of text classification of the reviews, logistic regression performs well with three class labels as well. A comparison of the accuracies can be seen in the graphs below.

For two class labels:



For three class labels:



Conclusion and Improvements:

The results produced via all the classification techniques are quite good in comparison to the human generated baselines. Logistic Regression performs the best followed by Random Forests and Naïve Bayes performs the worst when there are three class labels. The difference in accuracies between SVM, Logistic Regression and Random forests is very minimal. We can also see that all the accuracies of the classifiers is lesser when there are three class labels than the accuracies of the same classifier when there are only two class labels.

We have used unigrams as features. The performance of the classifiers might be better if we use bigrams. Using Tf-Idf to generate feature set instead of bag of words would have given better accuracies. A bigger training set would surely help in training the model better and predicting better. To determine sarcastic reviews, we can append Parts of speech tags. A combination of all the above methods would also improve the accuracy.

References:

https://www.cs.cornell.edu/home/llee/papers/sentiment.pdf

http://jmcauley.ucsd.edu/data/amazon/

http://fastml.com/classifying-text-with-bag-of-words-a-tutorial/

https://machinelearningmastery.com/gentle-introduction-bag-words-model/

http://ataspinar.com/2015/11/16/text-classification-and-sentiment-analysis/#SL_literature

https://www.sccs.swarthmore.edu/users/15/crain1/files/NLP_Final_Project.pdf