Aspect Based Sentiment Analysis

# Literature Survey:

Since the given problem was a text classification problem, I immediately started looking for papers that did aspect-based sentiment analysis using classification algorithms.

I found this [research paper](https://www.researchgate.net/publication/318857333_Aspect-based_sentiment_analysis_to_review_products_using_Naive_Bayes) written by [Mohamad Syahrul Mubarok](https://www.researchgate.net/profile/Mohamad-Syahrul-Mubarok) and Kang Adiwijay**a** explaining how aspect-based sentiment analysis works in detail in a simple yet professional manner.

The main takeaway from reading this and a few other papers were that all these models have 4 parts in them  
1. Aspect extraction – This step goes through each line and identifies the aspect term and returns all distinct possible aspects in a given statement.

2. Aspect polarity – For a given set of aspect terms, determine if the polarity of each aspect term is positive or negative

3. Aspect term categorization – This step categorizes the given aspect into a list of predefined aspect terms.

4. Aspect category polarity detection – Given a set of predefined aspect categories, this step finally determines the polarity (positive, negative or neutral) of each aspect category.

# Training approach

After reading a few more research papers, I started analyzing the dataset to start with the model building. Unlike the data from all research papers, the dataset had already pre-extracted aspect terms along with the text and polarity (0 – negative, 1- neutral, 2 – positive). Now since we already have the aspect term in the given dataset, the problem of categorizing the aspect term is reduced.

I started the training approach by just building a preprocessing function, which cleans the dataset of all unwanted data such as removing stop words, removing punctuation, tokenizing the text, lemmatization to get to the base word.

Now, I had a dataset that looked something like this:

After cleaning the dataset, I ran a simple Naïve Bayes Classification to predict the labels.

This model gave 62.345% accuracy, which is not good when it comes to classification problems.

After looking at how the Naïve Bayes model would work on the dataset without preprocessing the data, I tried to increase the accuracy with the same model by making some changes in the data.

Looking at the given dataset for a while, I saw that extracting aspect-based clauses or phrases are necessary to determine the polarity of the given aspect. The dataset had statements with both positive and negative sentiment in a given text. For example, the dataset had a line like:



So this is a compound sentence with 2 phrases in it.

The first part of the text describes how the user loves the notion as a tool, which is positive,

And in the second half of the sentence, the user says that the mobile takes way too long to load, which is negative. Feeding this statement to a basic Naïve Bayes model it gives “Negative” as output.

Since the aspect here on which the sentiment has to be analyzed is tool the given prediction is wrong since the user has a positive sentiment towards the tool.

Hence it was important to extract the phrases from the given text which had the aspect terms in it.

I wrote a function to extract clauses or phrases from a given sentence, using conjunctions like and, but, however, and simple punctuations like [‘,’,‘.’ ‘,@’ ‘!’ ‘&’ ‘….’ ‘/’].

After dividing a given sentence into respective clauses, it was important to extract the clauses with a given aspect which was predefined the data.

So, I wrote a preprocess function, which identifies the aspect from the given statement and extracts the clause with the given aspect in it.

Now the statement

C:\Users\dym1kor\AppData\Local\Microsoft\Windows\INetCache\Content.Word\IMG-20211229-WA0007.jpg

Became



This statement now has the positive aspect which the label is appropriate for.

It was now time to clean the given data further, like cleaning the dataset of all unwanted data such as removing stop words, removing punctuation, tokenizing the text and finally lemmatizing to get to the base words in the text.

The dataset now looks something like this:

Now since the testing was based only on positive and negative sentiments, I removed all the columns with the neutral sentiment.

# Approaches

My first thought was to train the dataset using Naïve Bayes which is a pretty powerful model to predict the classification.

Naïve Bayes is a conditional probabilistic model which works like this:

Given an instance to be classified, represented by a vector x = [x1, x2, x3…xn] representing N features, it assigns the instance to its probability.

Training the dataset on multinomial naïve Bayes classification, I got an accuracy of 79.855%.

Now that’s a big improvement from the previous classification, which just had an accuracy of

60%.

To increase the accuracy, I tried playing with the dataset by stemming words instead of lemmatization or

Detokenizing the dataset. All these methods only increased the model’s accuracy by .001%.

So now it was clear that I had to move on to other models and try different approaches.

Now I trained the given dataset with SVM (Support vector machine) which is a simple yet powerful model for classification problems.

Training the preprocessed dataset using SVM, it gave an accuracy of 82.23%.

Now that’s a small improvement from the multinomial Naïve Bayes model, but a good one.

After SVM, I tried classification with classification models like logistic regression, KNN, K means clustering and the Gaussian Bayes model. All these models gave an accuracy of around 79%, which is not better than the SVM model.

Finally, I tried a deep learning approach, by using the LSTM model to classify the given text.

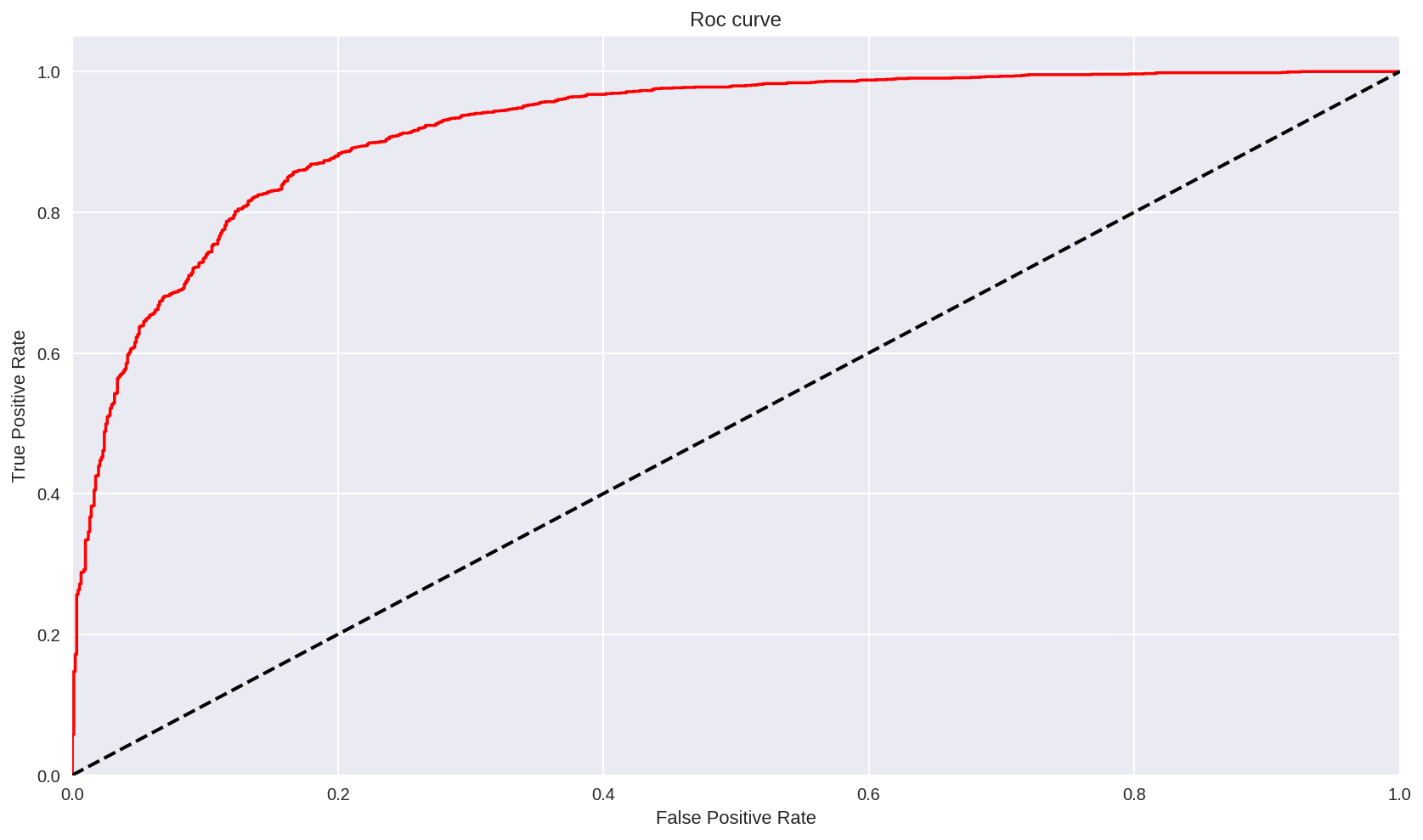
I chose LSTM in particular because LSTM is a model with gates such as input gate, forget gate, output gate, and cell state. My thought process was that the LSTM model would keep only relevant words that determine the sentiment of a text like good, better, worse, disappointing, and forget/discard all the words which are not necessary to predict the sentiment of the model.

Now after training the data using the LSTM network, it gave an accuracy of 76%, which was the lowest accuracy obtained compared to all the other models.

# Ablation Study

| No | Model | Accuracy |
| --- | --- | --- |
| 1 | Multinomial NB | 79.855%. |
| 2 | SVM | 82.23%. |
| 3 | LSTM | 76% |

ROC curve for SVM model



# Error Analysis:

After carefully analyzing the prediction, I observed that the model was predicting wrong sentiments for aspects that were not frequent in data. Since the data consists of a lot of singly repeated aspects, the error value is prominent. But this could be solved by either augmentation of the data or adding new data to the training set.

# Conclusion:

The given data works best with an SVM classifier, with around 83% accuracy.

Muhammad Dwi Aldhi