What is Sentiment Analysis?

It is the procedure to recognize and categorize opinions conveyed in text to decide the writer’s intention towards the topic of the post.

To accomplish this, we use algorithmic approach and try to classify all the records according to their respective category.

What makes reviews hard to classify?

Human Language is natural language not a formal language, it is difficult to analyze

For example

‘Perfume review: if you are reading this because it is your darling fragrance, please wear it at home exclusively, and tape the windows shut’ --very difficult to figure out using just positive or negative words.

‘This film should be brilliant. it sounds like a great plot, actors are first grade. However, it can’t hold up.’ -- final sentence says a lot about the review .

Automated sentiment analysis cannot understand sentiment in the context of business goals. -create new corpus based on specific words for machine to understand.

Approach:

**Text Preparation:** Cleaning the extracted data before analysis. Non-textual contents like hyperlinks, special characters, number that are irrelevant for the analysis are identified and eliminated.

* First the text data is uploaded to Python using pandas and a graph is plotted to know the number of records present in each class. It is found that nearly 60% of the data is marked as Negative sentiment. This is data with slightly Class Imbalanced.
* In order to do Sentiment Analysis there is need of just words but looking at the data it has noise like hyperlinks, number, special characters etc.
* During preprocessing, text has been converted to lower case, all the characters other than a-z are removed and words which have length greater than two and less than 15 are only preserved. Next the data is divided into train and test with .75 and .25 ratio this should be done before model building because it will give realistic prediction of how model will perform. We use model to learn on train data and predict the test data. With this procedure, one can know the performance of the model on unseen data i.e. how well a model is generalized.
* For an algorithm to do text analysis, raw data cannot be sent as input as scikit expects numeric input with fixed size. We need to take text and convert into vectors for that I used CountVectorizer which converts text into matrix of word counts. The role of fit is to learn the vocabulary of training data and transform is going to transform training data into document-term matrix with size 750\*4313 matrix where 750 are the unique records and 4313 are the vocabulary terms of train data.
* In order to **make a prediction**, the new observation must have the **same features as the training observations,** both in number and meaning. Hence, we will only transform the test data and the countvectorizer drops the words in the test data that are not seen during model training. countvectorizer is just for extracting features and building document-tern matrix. The real model building uses this matrix to learn and predict.

**Sentiment Classification:** In this step, Different models are tried out and I would like to explain Naïve Bayes model which learnt from train data and applied on test data. The model has classified records into positive, negative and neutral.

* Naïve bayes is a good one to start as it does very fast computations.
* Imported Multinomial naïve bayes from sklearn and fitted the model with transformed train data(X\_train\_dtm) with their respective class of each record(y\_train). (.fit) does the job of learning the matrix
* (.predict) is to predict the sentiment on transformed test data.
* For knowing the model performance, we have to compare the predicted class with the actual class. For this we use different measuring units for which we need to understand confusion matrix. For this document, I am going assume that the reader has knowledge on confusion matrix
* Now, we know from the data that the null accuracy is going to be 0.592 i.e. if we blindly predict every record as negative then we are going to be ~60% of the time correct. The model should beat the null accuracy and accuracy is not the only metric we have to focus for precision and recall.
* The confusion matrix generated from Naïve bayes is;

|  |  |  |
| --- | --- | --- |
| 159 | 4 | 0 |
| 35 | 29 | 0 |
| 20 | 2 | 1 |

Negative

Neutral

Positive

* From the matrix we can see that 159 are correctly classified as negative out of 163. This is because of having more records labelled as Negative and also there may be words that are present in three classes and their probability of classifying record as negative might be high.
* Only 29 are correctly classified as neutral out of 64 records. This is possible cause model learned more for negative classification and same goes for Positive labelling as there are few Positively labelled records in the dataset.
* The overall accuracy is 0.75 on test data which is it outperformed the null accuracy by 15%. Now the precision and recall are .88 & .59 respectively. This score shows that the model is more inclined in falsely predicting as positive i.e. it is classifying the records as negative whereas the records are neutral or positive.

**Findings and Output:** The main objective of sentiment analysis is to categorize the text into appropriate category based upon the words it contains.

Examining model for insights:This is to explore why and how naïve bayes has classified the records. We are going to find out which words are Negative, Positive and Neutral sentiment carrying words.

* Naïve bayes counts the number of times each word has occurred in respective class. I have separated each class word frequency and have combined them into a single pandas dataframe. Now we have the count of each word in each class where the records are the words and the columns are classes.
* Converting the values into percentages i.e. dividing each columns with its sum of frequencies. So this gives the approximate percentage of time a word appears in each class.
* Calculating negative ratio so as to find out which words have more impact for classifying a record to negative class and the same for positive class. Now with this we can say that the words like danny, prithvjit, pravir, meghana, shah etc have high positive ratio means they are highly satisfied with the services and would like to continue further or something like that (as I do not know the primary reason of chat).

The usage of logistic regression on data has yield similar results and there is still work needed to reduce the **bias** which in turn increases the overall model generalization.

The use of Random Forest has heavily overfitted the data and this shows the words used for training are sufficient to get the model to 75% but there are new words in test data that are not taken into consideration by countvectorizer.

I believe having more and equal proportion of data can help the model predict better and also the implementation of sentiment lexicons to find out the words which have positive and negative meaning cause naïve bayes is classifying records just based on frequency of words. Apart from this, create a corpus of words that are specific to organization as they might appear in all the classes eliminating those unnecessary words will contribute in improving the model prediction.

One should really understand the text, clean it in business perspective, understand the model and the tuning parameters to decide which one are probably worth working on. After all this is text and a message can be conveyed using different types of words, phrases and this makes text classification challenging.

**A bad model on clean data is better than a good model on noisy data.**

**Future potential area:** As most of the text is sent to data lakes, analyzing and processing that amount is inconvenient. Based on the business problem one can extract positive or negative reviews and analyze them and plan their production, supply chain or improve the product and tailor it to customer expectation etc.

This can also be used for elections, the opinion of people on social media about the new changes in rules (like the GST).

Thanks for going through the document and I would like to explain further if we can have personal conversation as I did not elaborate every line of code and few concepts.