Sentimental Analysis is an Integral part of Text Analytics. Sentimental analysis is a process of computationally identifying and categorizing opinions based on the choice of word usage, whether it is positive, negative or neutral.

One such analysis is calculating if one particular request is polite or not. There are various strategies used to determine the politeness of a request.

Predicting politeness:

In this project, we have used SVM Modelling, which is a supervised learning algorithm. For this we need to have a data set which is already manually labelled whether that statement is positive, negative or neutral.

In order to prepare the data, we have to first

1. Label the data
2. Generate vocabulary
3. Create a document term matrix.

So in our code, ‘create\_matrix’ function creates the document term matrix. In a simple example, creation of a document term matrix is given below

Our goal is to predict if the statement is impolite or polite. To simplify, I have written each sentence in the words of impolite and polite. We should classify it as ‘+1’ if it is polite and ‘-1’ if it is impolite

Polite, 1

Impolite, -1

Polite, Impolite, Polite,1

Impolite, Impolite, Polite, -1

Impolite, Impolite, Impolite, -1

Polite, Polite, Polite,1

Now, we look at each of our sentence and generate vocabulary. We first write each of it in the alphabetic order i.e., Impolite and Polite. We begin each line with the class of the sentence +1 or -1. Next comes the index of the word and we add a colon, then the number of times the word appears in the sentence.

|  |  |
| --- | --- |
| +1 | 2:1 |
| -1 | 1:1 |
| +1 | 2:2 1:1 |
| -1 | 1:2 2:1 |
| -1 | 1:3 |
| +1 | 2:3 |

This is how we create a document term matrix of the whole data. Then we divide our data into training data and testing data. We create a document term matrix in the training data and make a model out of it. Then we test the data and predict our results.

There are few features to include in our code to improve our accuracy

1. N-Grams: An n-gram model models sequences, notably natural languages, using the statistical properties of n-grams. N-gram models are widely used in statistical [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing). In [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition), [phonemes](https://en.wikipedia.org/wiki/Phonemes) and sequences of phonemes are modeled using a N-gram distribution. For parsing, words are modeled such that each n-gram is composed of n words. For [language identification](https://en.wikipedia.org/wiki/Language_identification), sequences of [characters](https://en.wikipedia.org/wiki/Character_(symbol))/[graphemes](https://en.wikipedia.org/wiki/Grapheme) (e.g., [letters of the alphabet](https://en.wikipedia.org/wiki/Letter_(alphabet))) are modeled for different languages.
2. Term Frequency – Inverse Document Frequency: In information retrieval, tf–idf, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval, text mining, and user modeling.

The steps we used in this project

1. Read the data into R
2. Remove stop words also use an extensive list of stop words by creating your own custom stop word list
3. Divided the data into n – grams (3 in this project)
4. Remove sparse terms which have appeared for very less times in the document
5. Convert our test label into integers (Positive 🡪1, Impolite 🡪-1 , Neutral 🡪 0)
6. Create a document term matrix of the input file.
7. Create a container with the training data and train the Model in SVM
8. Now define the test data and create a prediction document matrix
9. Now we check our accuracy of the model using Confusion Matrix
10. Now we convert the predicted label back to our original label (Positive, Negative and Neutral)