***Emotion Detection using Text***

***Data Mining Project Report***

Sai Srujana Gunda | Dhrishya Menon | Brunda Uppalapati

***Table of contents:***

* Introduction------------------------------------------------------------------------------------ 2
* Problem Statement---------------------------------------------------------------------------- 2
* Data--------------------------------------------------------------------------------------------- 2
* Methodology ---------------------------------------------------------------------------------- 3
* Pre-processing -------------------------------------------------------------------------------- 4
* Feature Extraction ---------------------------------------------------------------------------- 4
* Model ------------------------------------------------------------------------------------------- 5
* Evaluation-------------------------------------------------------------------------------------- 6
* Challenges -------------------------------------------------------------------------------------- 7
* Scope-------------------------------------------------------------------------------------------- 7
* Appendix---------------------------------------------------------------------------------------- 7
* References -------------------------------------------------------------------------------------- 8

**Introduction:**

The Internet is a place with the information being stored majorly in the textual format. It contains large collection of documents, information from the blogs, user reviews of products, movies etc. This information provides accessible and plentiful data that can be analysed for range of applications. These data can be analysed using emotion to define the feeling of a given text at the document, sentence or feature/aspect level.

Emotions may be expressed in many ways such as facial expressions, gestures, speech and text. Emotion Detection in text is the newer area of research interest in the field of text analysis. It is essentially a problem of content – based classification involving various concepts from the domains of Natural Language Processing and machine learning.

Emotion Detection is closely related to sentiment analysis. Sentiment analysis aims to detect whether a given text is positive, negative or neutral; whereas emotion detection tries to detect and recognise the types of feelings from the given sentence/phrase in a text document such as anger, sad, joy, surprise, fear, love, hate etc. This can be used to understand the feeling of a customer on a product or a brand and improve product/brand with the necessary updates. It is very useful in case of human – machine interaction. Detecting an emotion from text is becoming important from the application’s point of view. Emotional system can be formally classified through an emotional hierarchy in six classes at primary level which are Love, Joy, Anger, Sad, Surprised, Fear. For our analysis, we have considered only five emotions of Joy, Sad, Surprise, Fear, and Anger.

**Problem Statement:**

Sentiment analysis of a text can only say if a particular sentence conveys positive or a negative polarity. If we can classify the text furthermore based on the emotion of the content, it can be used by a product/brand/public figure to make necessary improvements in their respective fields. With the help of this information, the perspective of the users towards that brand can be improved in a positive way.

Chatbots are being extensively used for providing services to the customers in various sectors. Identifying the emotion of the user can help them analyze how well they are able to meet the requirements of the customers and make necessary changes to the system. In case the emotion of the customer turns out to be either Angry/Sad they can connect the customer to an executive to take quick actions.

**Data:**

We have grouped below four datasets.

1. **Novel Dataset:** This dataset consists of different phrases from a novel. These phrases have been labelled according to their emotion. This dataset has small number of records.
2. **Twitter Dataset:** This dataset was collected using Twitter public streaming API. The collected tweets were automatically labelled using the emotion hashtags at the end of each tweet. It consists of 20K records, as this dataset have been already labelled, we have only considered 5 labels namely, **Joy, Sad, Surprise, Fear, and Anger.**
3. **Kaggle Dataset:** It consists of 4 Million records of labelled data consisting five emotions. We randomly selected 6000 records of each emotion for the model. So that every emotion is normalised. We have reduced the dataset from 2M to 25K due to data processing and modelling constraints by system.
4. **Custom Dataset:** We have also added few of our custom data by referring through blogs, news feeds and custom sentences for initial analysis which we felt as necessary in the dataset.

After collecting the data, we have normalized the emotions to **Joy, Sad, Surprised, Angry, and Fear** so that the data from all the sources will have the same labels.

**Methodology:**

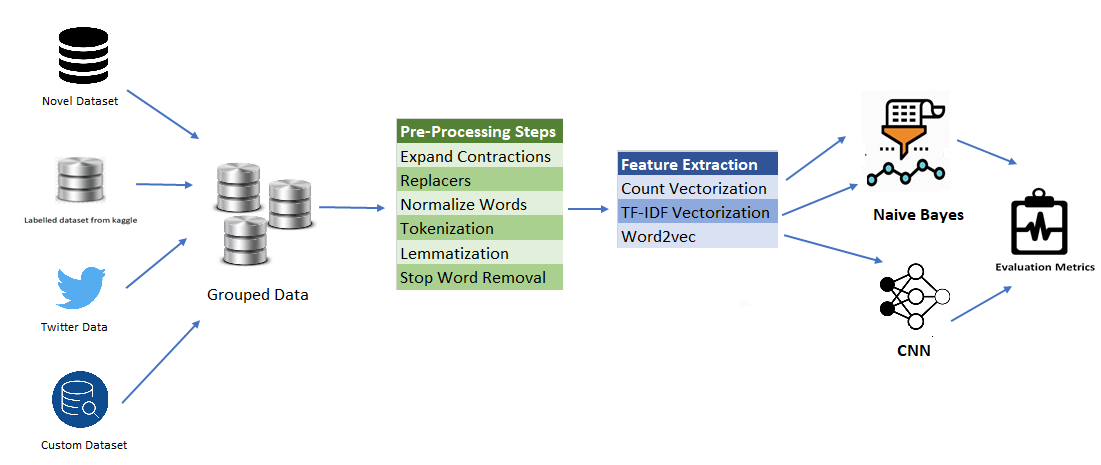
We have followed below mentioned approaches as a part of word embedding which is one of the language modelling and feature learning technique where words or phrases from the vocabulary are mapped to vectors of real numbers.

1. **Frequency based Word Embedding:**

In this embedding, we have used Count Vector (Bag of Words) and TF-IDF vector (Term Frequency – Inverse Document Frequency) in which we initially measure how frequently a term occurs among all sentences and determine how important a word is to predict the emotion of a given sentence.

1. **Prediction based Word Embedding (Word2Vec):**

In this embedding, all the words are converted into vector from in which similar words share the same spatial position. These word embedded vectors are fed to a Convolution Neural Network (CNN) model which trains the model by processing the vectors thereby, providing the predicted emotions.

 **Fig 1: methodology to predict emotion detection from text**

**Pre-processing**:

It can be inferred as converting data to a format that a model can understand. Below is the flow of data pre-processing before loading the data into the model:

* **NormalizeWords**: Created a function which will remove non-ASCII values. There may be emoji’s, punctuation marks etc. All these characters except letters are removed from the sentences and are converted to lower case letters.
* **RepeatReplacer**: Replaced repeated words with their respective root words.

Example: Words like Happyyyyyyyyyy is replaced with happy

* **ExpandContractions** : Expanded the contracted words, so that the true meaning of the word will not be changed, and true emotion of the sentence can be captured.

Example: Don’t is replaced with Do not. Haven’t is replaced with Have not etc.

* **Stop word Removal:** A stop word is a commonly used word that a search engine has been programmed to ignore so that we can save processing time, space and avoid giving weights to non-significant terms. For instance, words like is, an, the, and so on are present multiple times in a document which don’t contribute significantly in predicting the emotion. Hence, we have removed these stop words. However, negative words like “Not” have significance as they might convey negative emotion (not happy). Hence, we have removed possible negative words from the list of stop words.
* **Tokenization**: We have used Tokenizer to convert the words in a sentence into tokens so that it can be used as an input to a Lemmatizer.
* **Lemmatization**: It is the process of grouping the changed form of words into root word based on the context which helps in providing term frequency for better prediction of the label. As a part of pre-processing, we have tokenized the sentences and tagged with their respective parts of speech and then converted them into their root words accordingly.

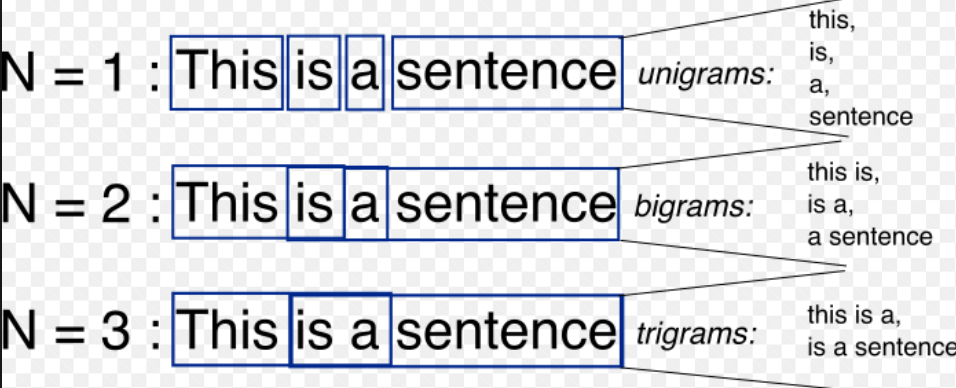
Example: Words go, going, gone belong to same root word “go”. Hence, one sentence might contain word “going” in it and other might contain “go” in it. By converting them to their respective root word “go”, we will be able to get rid of unnecessary weights for non-significant words.

**Feature Extraction:**

After Pre-processing the data, we have extracted the features from text data. Feature Extraction is a dimensionality reduction process, where an initial set of raw variables is reduced to more manageable groups for processing, while accurately and completely describing the original data. We have used Bag of words model, TF-IDF model and Word2vec to extract features.

* **Bag of words:** This model is simple in understanding and implementation. The specific strategy of tokenization, counting and normalization is called Bag of Words representation. Sentences in this model are described by word occurrences while completely ignoring the relative position information of the words in the document. Each word or token is called a gram.

**Grams:** Words are basic meaningful elements with the ability to represent a meaning when written in a sentence. Below is the picture of difference between different grams the system reads.



We have used bigram in our model. As we are analysing a sentence, there is possibility that words such as ‘not good’, ’not working’ etc can change the complete meaning of the sentence if we consider only ‘good’. By using a bigram in the model, the words ‘not good’ will be considered as one and the system reads it as one word so that the emotion will not be altered and predicted correctly.

By considering Bag of Words model taking into consideration bigrams, we have built a vectorizer which will be given as an input to a classifier.

* **TF-IDF Vectorisation:** TF-IDF stands for Term frequency and Inverse document frequency. Term Frequency (TF) measures how frequently a word occurs in a document. There are few words which occur many times in a document, however have less information in predicting emotion. Hence, IDF measure is used to decrease the weight for commonly used words and increase the weight for words that are not used much in a set of documents. We have used TF-IDF model with bigrams to get the vectorizer.
* **Word2vec:** Word2vec is a group of related models which are used to produce word embeddings. These models are two-layer Neural networks that are trained to reconstruct linguistic context of words. Usually word2vec takes large data as input and produces a vector space which are typically of several hundred dimensions. Each word in the data is assigned a corresponding vector in the space. These word vectors are positioned in the vector space in such a way that words which share common context are near to one another in the space.

**Continuous Bag of Words Model:** In continuous bag of words, the current word is predicted by the model using a window of surrounding words. This method takes context of each word as the input and tries to predict the word corresponding to the context.

**Skip-Gram Model:** The Skip gram model takes every word in the data and takes one-by-one the words that surround it within a defined ‘window’. This is then feed to a neural network that after training will predict the probability for each word to appear in the window around the focus word.

In our model after finding the word vectors, we have concatenated the CBOW model and Skip-Gram Model representations to construct a dictionary so that necessary features can be extracted.

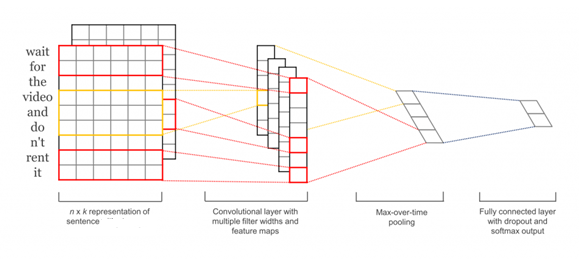
**Models:**

Once the Pre-processing is done, the vector representations of the words will now be loaded into the model. We have considered two models for predicting the emotion.

* **Multinomial Bayes:** Multinomial Naive Bayes is a specialized version of Naive Bayes that is designed more for text documents. Whereas simple naive Bayes would model a document as the presence and absence of words. Multinomial naive Bayes explicitly models the word counts and adjusts the underlying calculations to deal with in. It estimates the conditional probability of a word given a class as the relative frequency of term (t) in documents belonging to class (c). The variation considers the number of occurrences of term “t” in training documents from class (c), including multiple occurrences.

We have used the vector representations of Bag of Words model and TF-IDF as inputs to the Multinomial Bayes model. We have divided the entire data into 80% of training data and 20% as test data. After training the model we have checked how well the model performs using test data.

* **Convolutional neural networks (CNN):** CNN’s are very effective deep learning models for analysing the given inputs and classifying. The output from the word2vec model are given as input to the CNN. CNN’s use a sliding window called kernel or filter to perform convolutions over the input matrix of words as shown below.



In our model, for creating embedding matrix that fits all the words, we have counted the length of all the words in the dataset which includes both test and training set and created an embedding matrix of that size so that all the words would fit into the vector and words which are less than the maximum length will be padded with zeros. We have used 100 filters for features detection with stride size of 1 which covers the n gram concept. The convolution layers use non-linear activation functions like ReLU or tanh. We have used one hidden layer having 256 nodes with ReLU as our activation function. To fit the multiclass labels, we had to convert the labels into oneHotLabelVectors

After the convolutional layers, we need to perform pooling operation. Pooling layers subsample the input from convolutional layer. The most common way to do pooling is to apply a max operation to the result of each filter. Pooling is necessary because it provides a fixed size output matrix, which is required for classification. We have used max pooling in our model.

After the pooling is done we have given our output to a fully connected layer with SoftMax output for classification.

**Evaluation Metrices:**

|  |  |  |
| --- | --- | --- |
|  | **Multinomial Naïve Bayes** | **CNN** |
| **Test Accuracy** | 0.89 | 0.97 |
| **Precision** | 0.90 | 0.913 |
| **F1-Score** | 0.899 | 0.912 |
| **Recall** | 0.899 | 0.912 |
| **ROC Score** | 0.93 | 0.94 |

**Interpretation:**

The results of both the model looked almost decent with CNN on higher side. With the Test accuracy of almost 97% the model was better able to predict the emotions of test set within the dataset. With the precision of 90% in both models, 90% of precited emotions were predicted correct. We consider F1 score if we have uneven class distribution. We got the precision of 90% which means that of all the instances with emotions, labelling was properly done. The reason for CNN accuracy to be almost same as Naïve Bayes is because of lack of the huge data to train the model as the computer processing capabilities are limited.

**Test results of Twitter data and Custom Data:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Evaluation metrics** | **Twitter dataset** | | **Custom dataset** | |
| **Multinomial NB** | **CNN** | **Multinomial NB** | **CNN** |
| **Test Accuracy** | **0.71** | **0.798** | **0.89** | **0.97** |
| **Precision** | **0.68** | **0.88** | **0.90** | **0.913** |
| **F1-Score** | **0.66** | **0.788** | **0.899** | **0.912** |
| **Recall** | **0.63** | **0.76** | **0.899** | **0.912** |

To ensure that model works better on all datasets, we tested the model on Twitter and Custom datasets. We found that Accuracy on Twitter dataset was around 79%. As the twitter data set contains tweets from various users which is not in standard English. Hence, the data was not completely cleaned as a result, the accuracy was low with both the models.

When we tried with Custom dataset which contains data from various blogs, Books and Kaggle. We found that the accuracy was increased to 97% which was pretty good as the data contains texts of standard English. We can see that CNN model accuracy was less than Multinomial NB as deep learning model need large amount of data to train.

A graph of a function

Description automatically generatedA graph of a function

Description automatically generated with medium confidence

**Fig .2: ROC curves of Multinomial naive Bayes and CNN models respectively(refer appendix for ROC curves of each emotion)**

**Challenges:**

The biggest challenge in emotion detection is it is dependent on context within a text. A Sentence or a Phrase can have the emotion of anger/disgust without using the word anger/disgust.

1. We found that there were many sentences which can be easily interpreted by human brain when compared with model predicted results.
2. We also found that text itself is not enough in predicting emotion without considering the context or the intensity of the sentence. Hence, emotion detection from speech makes more sense.
3. We have also found that the model may provide wrong results for complex sentences which may contain negative words even after considering bigram as we don’t have enough sentences with negative words in our training set.
4. We have removed punctuations like exclamation (!), question marks (?) which would have helped us in providing much more information in predicting emotions, as exclamation might help in predicting surprised or shocked emotions.

**Scope:**

As part of future scope, we want to extend this project in predicting the emotion from a speech which would add up to our existing model and then combine both speech and text in better prediction of the given sentence. Thereby, we feel that we can overcome the challenges which we faced during our text analysis that would help in increasing the accuracy of the model prediction.

**Appendix:**

**Multinomial Naïve Bayes (Joy, Sad, Surprised, Angry, and Fear)**

A graph of a function

Description automatically generated**A graph of a function

Description automatically generated with medium confidence**

A graph of a function

Description automatically generated**A graph of a function

Description automatically generated**

A graph of a function

Description automatically generated

**CNN (Joy, Sad, Surprised, Angry, and Fear)**

A graph of a function

Description automatically generated with medium confidence**A line graph with a point

Description automatically generated**

A graph of a function

Description automatically generated with medium confidenceA graph of a function

Description automatically generated with medium confidence

**A graph of a function

Description automatically generated with medium confidence**

**References**:

1. <https://www.quora.com/How-does-multinomial-Naive-Bayes-work>
2. <https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html>
3. <https://medium.com/data-science-group-iitr/word-embedding-2d05d270b285>
4. <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>
5. <https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-11-cnn-word2vec-41f5e28eda74>