Emotion Classification on Twitter Data

Titli Sarkar
Department of Computer Science
University of Louisiana at Lafayette
Lafayette LA, USA
C00222141@louisiana.edu

Dr. Aminul Islam
Department of Computer Science
University of Louisiana at Lafayette
Lafayette LA, USA
aminul@louisiana.edu

***Abstract*— Twitter is one of the most popular social media nowadays used for expression people's opinions. Twitter offers corporate and business organizations a fast and effective way to analyze customers' perspectives toward the critical to success in the market place. This paper addresses the problem of classifying emotion from tweets. Four basic types of emotions: anger, fear, joy, and sadness are identified. Three types of classifier models are used: Gaussian Naïve Bayes, Bernoulii Naïve Bayes and Random Forest to classify sequential data with semantic meaning. GloVe [15] word embeddings are used to represent the corpus as word vectors with semantic meaning. After preprocessing with sentence tokenize and stopwords removal, feature selection is done using GloVe. The word embeddings of each tweet and class label of each of them are passed into the classifier model for training and testing purpose. 59% best predictive accuracy is achieved with Gaussian Naïve Bayes, 59% best predictive accuracy is achieved with Brenoulii Naïve Bayes and 61% best predicted accuracy is achieved by Random Forest on test dataset. Different feature selection methods in combination is needed to be used and different classifier models must be considered to increase the accuracy of the model.**

Keywords— —Twitter Emotion Classification; GloVe word embeddings; Naïve Bayes; Natural Language Processing

I. INTRODUCTION

Emotions play key role in human intelligence, rational decision making, social interaction, perception, memory, learning, creativity, and more. Nowadays, social media provides a platform for business which allows to connect the owner with their customers to advertise or to know the customer's perspective of the products and services. Social media plays a significant role in politics as the crowd's reaction is one of the most important metric for the politicians to decide their future steps. Sentiment analysis is one of a most popular subtopic of natural language processing and text mining that deals with the automated discovery and extraction of knowledge about people's sentiments, evaluation and opinions from textual data such as personal blogs, review websites and customer feedback forms. Sentiment analysis has received significant interest in recent times because of its practical usage and application in today's environment. Twitter is one of the most popular social media for expressing emotions. The goal of Twitter emotion classification is to automatically determine the emotion expressed in a tweet. After analyzing the trend of the basic emotions present in popular tweets, we found out people have expressed four types of basic emotions: anger, fear, joy and sadness. We propose a novel method of classifying the emotion for tweets using Naïve Bayes and Random Forest.

We are thankful to SemEval'2018 organizers for providing the data for the Task 1 which is Affect in tweets, falls under the category Affect and Creative Language in Tweets. The dataset is labelled and divided in three parts: Train Data, Development Data and Test Data. We have processed all the data in each of the train, development and test set before feeding them to the neural network as a mandatory part of any natural language processing problem. We have sentence tokenized and word tokenized each tweet, stripped punctuations and removed stop words from them using python NLTK library. After preprocessing is done, we have represented each word of the tweets as word embedding using GloVe as feature selection. To make the tweet corpus equidimensional, zero padding is done for each tweet. The preprocessed corpus is now fed into different classifiers. We have tried Gaussian Naïve Bayes, Bernoulii Naïve Bayes and Random Forest classifiers and compared the results. The result shows that Random Forest works slightly better than Naïve Bayes in our case.

This paper is organized as follows. In section II, the related work is reviewed. Section III describes the methods and ideas which includes the data preparation, feature selection and model building steps. In section IV, results and description of results are included. Finally, in section IV, we drew the conclusion.

II. RELATED WORK

S. Radha Krishna et.al[1] studied different spectral features such as MFCC, pitch chroma, skewness and centroid for emotion recognition. The emotions considered in this study are Fear, Anger, Neutral, and Happy. The system is evaluated for various combinations of spectral features. It is established that the combination of MFCC and skewness gave better recognition performance when compared with other combinations. These experiments are conducted and evaluated using Gaussian Mixture models (GMMs).The data base used in this study is Telugu emotion speech corpus (IIIT-KGP).

Taner Danisman et.al [2] studied, automatic classification of anger, disgust, fear, joy and sad emotions in text.The study was conducted on ISEAR (International Survey on Emotion Antecedents and Reactions) dataset. For the

classification they have used Vector Space Model with a total of 801 news headlines provided by "Affective Task" in SemEval 2007 workshop which focuses on classification of emotions and valences in text. They have compared their results with ConceptNet and powerful text based classifiers including Naïve Bayes and Support Vector Machines. Their experiments showed that VSM classification gives better performance than ConceptNet, Naive Bayes and SVM based classifiers for emotion detection in sentences. An overall F-measure value of 32.22% and kappa value of 0.18 for five class emotional text classification on SemEval dataset which is better than Navie Bayes (28.52%), SVM (28.6%)was obtained.

Swati D. Bhutekar et.al [3] presents a methodology to extract emotion from the text at real time and add the expression to the textual contents during speech synthesis. This paper also focuses on implementation of creation of Corpus, emotion recognition module etc. In text analysis, all emotional keywords and emotion modification words are manually defined. The test was carried out on set of textual sentences and preliminary rules written for 34 different emotions. These rules are used in an automated procedure that assigns emotional state values to words. These values are then used by speech synthesizer to add emotions to speech & input sentence. Pitch detection algorithm has been implemented for pitch recognition. The system is language dependent. Changqin Quan et.al[4], make an analysis on sentence emotion based on emotion words using Ren-CECps (a Chinese emotion corpus). Some classification methods (including C4.5 decision tree, SVM, NaiveBayes, ZEROR, and DecisionTable) have been compared. Then a supervised machine learning method (Polynomial kernel method) is proposed to recognize the eight basic emotions (Expect, Joy, Love, Surprise, Anxiety, Sorrow, Angry and Hate). Using Ren-CECps, we get the emotion lexicons for the eight basic emotions. Polynomial kernel (PK) method is used to compute the similarities between sentences and the eight emotion lexicons. Then the experiential knowledge derived from Ren-CECps is used to recognize whether the eight emotion categories are present in a sentence. The experiments showed promising results. Ali Houjeij, Layla Hamieh et.al [5] design ed a system that adopts a novel approach for emotional classification from human dialogue based on text and speech context. Their main objective was to boost the accuracy of speech emotional classification by accounting for the features extracted from the spoken text. The proposed system concatenates text and speech features and feeds them as one input to the classifier. The work builds on past research on music mood classification based on the combination of lyrics and audio features. The innovation in our approach is in the specific application of text and speech fusion for emotion classification and in the choice of features, Furthermore, in the absence of benchmark data, a dataset of movie quotes was developed for testing of emotional classification and future benchmarking. The comparison of the results obtained in each case shows that the hybrid text-speech approach achieves better accuracy than speech or text mining alone.

Amira F. El Gohary et.a l [6] are concerned with the automatic detection of emotions in Arabic text. This construction is based on a moderate sized Arabic emotion lexicon used to annotate Arabic children stories for the six basic emotions: Joy, Fear, Sadness, Anger, Disgust, and Surprise. Their approach achieves 65% accuracy for emotion detection in Arabic text.

Shadi Shaheen, Wassim El-Hajj et.alk [7],proposed a framework for emotion classification in English sentences where emotions are treated as generalized concepts extracted from the sentences. They generated an intermediate emotional data representation of a given input sentence based on its syntactic and semantic structure. They then generalized this representation using various ontologies such as WordNet and ConceptNet, which resulted in an emotion seed called an emotion recognition rule (ERR). Finally, a suite of classifiers are used to compare the generated ERR with a set of reference ERRs extracted from a training set in a similar fashion. The used classifiers are k-nearest neighbors (KNN) with handcrafted similarity measure, Point Mutual Information (PMI), and PMI with Information Retrieval (PMI-IR). When applied on different datasets, the proposed approach significantly outperformed the existing state-of-the art machine learning and rule-based classifiers with an average F-Score of 84%.

III. Methods and Ideas

In this work, we present LSTM and BiLSTM based neural network model which determine the emotion expressed in a tweet, given an arbitrary tweet. We get the output of the neural network in four output neurons and select the neuron with highest score as the output emotion. This work can be extended to predict emotion intensity also if we train the model with emotion labels as well as intensity scores.

A. DataSet

The dataset we have picked for usage from EmoInt 2018 task, have a total 8566 tweets in English language for training and validation. We have picked total 2000 tweets mixed tweets from each of the anger, fear, joy and sadness testing dataset in English language to use as our test data. We have make sure the dataset is equibalanced in each type of emotions by picking 500 tweets for each emotion testing data. Each of our datasets are divided in three parts: training data, development data and test data. In each of the set, we have separate files for anger, fear, joy and sadness. Each of the files has labelled data in the form "ID, Tweet, Affect Dimension, Intensity Score". The field Affect Dimension has four values: *anger, fear, joy, sadness.* We have annotated or labelled the emotions as anger - 1, fear - 2, joy - 3, sadness - 3 and combined the development data with the training data together to be used as training data. The reason of including the development data in training data is to give the model a little more data for learning such that it can make better prediction on test data. As our task is classification task, we

have combined tweets for all the emotions together in a single file for each of training and testing purpose.

B. Preprocessing

Preprocessing is a crucial step in natural language processing. Our data is available in raw form. It need to be processed in a form that is free from noise and hence does not create any problem for finding the word mbedding before feeding into the neural network. Following is the sequence of steps:

1. Tokenization:

Tokenization is the process of replacing data with unique identification symbols that gets rid of the noise but retain all the essential information about the data without compromising its meaning. It is usual practice in natural language processing and machine learning that we should split each paragraph into sentences and each sentence into words. We have used python NLTK tokenizer to break tweets which have more than one sentence to individual sentences and break each individual sentence into individual words.

2. Removing URL's:

Tweets often contain links to other websites. This is the case when these websites contain content, referencing which the tweet was posted. These are just unwanted noise and contribute in no positive way to classification and hence must be removed. This architecture uses regular expressions to remove any URL that is present in tweets.

3. Removing @'s:

The sign @ allows users to mention other users in their tweets. Like "@carljones very happy to hear your story!" tweet mention "carljones" which does not have any useful meaning to emotions, henceforth is removed.

4. Removing hashtags:

Sometimes people use normal words preceded by hashtags in sentences. For example, "Finally time to eat some #cake!". These sentences when tokenized are broken down along with the hashtag, and the words "cake" and "#cake" have different word representations. One must remove the hashtag punctuation symbol and interpret it as a plain word. This again is done by using regular expressions.

5. Strip punctuations:

We are handing with emotions; therefore it is usual that the tweets will have exclamation marks (!) and question marks (?). We have removed punctuations using python NLTK library in built method.

6. Removing Numbers:

Numbers does not have any importance when detecting emotions in text. In most cases, they simply convey some time or date or quantity and are totally irrelevant for in the current domain. They simply add noise and hence must be removed. Again regular expressions are employed to replace them with blanks.

7. Removing Stop Words:

Stop words are a set of commonly occurring words in any language. They typically are prepositions, conjunctions and determiners like "the", "a", "an", etc. Python NLTK library is used to remove stop words.

8. Lowercase Conversion:

The last step in pre-processing is to convert all the text to lowercase. It is important because it ensures that the word vector of each word is not affected by their case. This helps maintain uniformity throughout in the next step, where we are finding word embedding of each words in the tweet.

C. Feature Selection

We are working with textual sequential data which have some semantics, therefore, we need to retain the semantics of the data in our representation. We can use pre-trained models which have been trained on very large text corpus which have been derived from various sources like news, Twitter, Wikipedia etc. Advantage of using a custom trained model is that they learn words in the domain of application. For example, if one trains a model based on text corpus from Twitter, words will be learned in the domain of Twitter. Nowadays, a word embedding model called GloVe [15], developed by Standford NLP group is the most popular word embedding model for word representation for Global Vectors, because the global corpus statistics are captured directly by the model. Once the GloVe model is generated, we have used GloVe model to generate word embeddings for each of the tweets of training, development and test data set which are cleaned by several preprocessing steps, as a vector of features. This is done as follows:

1. Loading GloVe Model:

The pre-trained word vectors for GloVe Twitter are downloaded and saved in a directory. We need to load those pre-trained vectors as word2vc format for using in our python code. We have converted the GloVe file to word2vec format using python genism library. Then, using the same module, we have loaded the pre-trained model in word2vec format.

2. *Finding Word Embeddings:*

After the preprocessing step, we have broken each tweet into a list of separate words. Now we find word embedding vector for each word in each tweet using the GloVe model and represent them as a list of word embedding vectors. But one problem is the variation in length of tweets i.e. each tweet has different number of words and the corpus is not equidimensional. We must consider the length of the longest tweet and make others same length by adding zero *padding* to them. This work is done using python Keras module. The result is a vector of a dimension (differs in different models) for each tweet.

At this stage, we have equidimensional corpus for training and one-hot encoded labels. These will be used to train the neutral network. For custom trained vectors, the models are trained using nearly 9000 tweets. They will be tested against pretrained vectors of dimension 100 for 2000 test data which includes tweets of all kinds of emotions in a balanced manner. We have created a train dataset by concatenating the word embedding of the tweets present in the training dataset and the word embedding of the tweets present in the development dataset. Similarly, we have created a train label set by concatenating the emotion annotated labels in the train dataset and the emotion annotated labels in the development dataset. These two are served as training corpus and training labels for the neural network models.

D. Model

We have use Gaussian Naïve Bayes, Bernoulii Naïve Bayes and Random Forest as our models for building classifier for this specific problem of Twitter emotion classification.

a. Gaussian Naïve Bayes:

Naïve Bayes classifier [9] is based on the so-called Bayesian theorem with the naïve assumption of independence between every pair of features. This classifier in spite of the apparently oversimplified assumptions has worked quite well in many real world situations. It is very fast and has a good performance, better in some cases than more sophisticated methods. The Naïve Bayes model with Gaussian is equivalent toa mixture of Gaussians (GMM) with diagonal covariance matrices. The main advantages of this classifier are the conditional independence assumption, which helps to obtain a quick classification and the probabilistic hypotheses (results obtained as probabilities of belonging of each class). Bayesian classification is based on Bayes' theorem. Bayes' theorem is named after Thomas Bayes, a nonconformist English clergyman who did early work in probability and decision theory during the 18th century. Let X be a data tuple. P(H/X) is the posterior probability, or a posteriori probability, of H conditioned on X. In contrast, P(H) is the prior probability, or a priori probability, of H. Similarly, P(X/) is the posterior probability of X conditioned on H. P(X) is the prior probability of X. Bayes' theorem is useful in that it provides a way of calculating the posterior probability, P(H/X), from P(H), P(X/H), and P(X).
Bayes' theorem is

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}.$$

- **Gaussian Naïve Bayes:**

Naive Bayes can be extended to real-valued attributes, most commonly by assuming a Gaussian distribution. This extension of naive Bayes is called *Gaussian Naive Bayes*. Other functions can be used to estimate the distribution of the data, but the Gaussian (or Normal distribution) is the easiest to work with because you only need to estimate the mean and the standard deviation from your training data.

Representation for Gaussian Naive Bayes

With real-valued inputs, we can calculate the mean and standard deviation of input values (x) for each class to summarize the distribution. This means that in addition to the probabilities for each class, we must also store the mean and standard deviations for each input variable for each class.

Learn a Gaussian Naive Bayes Model From Data

This is as simple as calculating the mean and standard deviation values of each input variable (x) for each class value.

*mean(x) = 1/n * sum(x)*

where n is the number of instances and x are the values for an input variable in your training data.

We can calculate the standard deviation using the following equation:

*standard deviation(x) = sqrt(1/n * sum(xi-mean(x)^2))*

This is the square root of the average squared difference of each value of x from the mean value of x, where n is the number of instances, sqrt() is the square root function, sum() is the sum function, xi is a specific value of the x variable for the i'th instance and mean(x) is described above, and ^2 is the square.

Make Predictions With a Gaussian Naive Bayes Model

Probabilities of new x values are calculated using the Gaussian Probability Density Function (PDF). When making predictions these parameters can be plugged into the Gaussian PDF with a new input for the variable, and in return the Gaussian PDF will provide an estimate of the probability of that new input value for that class.

*pdf(x, mean, sd) = (1 / (sqrt(2 * PI) * sd)) * exp(-((x-mean^2)/(2*sd^2)))*

where pdf(x) is the Gaussian PDF, sqrt() is the square root, mean and sd are the mean and standard deviation calculated above, PI is the numerical constant, exp() is the numerical constant e or Euler's number raised to power and x is the input value for the input variable. We can then plug in the probabilities into the equation above to make predictions with real-valued inputs.

- **Bernoulii Naïve Bayes:**

In the multivariate Bernoulli event model, features are independent Booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks where binary term occurrence features are used rather than term frequencies. If x_i is a boolean expressing the occurrence or absence of the *i*'th term from the vocabulary, then the likelihood of a document given a class C_k is given by

$$p(\mathbf{x} \mid C_k) = \prod_{i=1}^{n} p_{ki}^{x_i}(1 - p_{ki})^{(1-x_i)}$$

where p_{ki} is the probability of class C_k generating the term x_i. This event model is especially popular for classifying short texts. It has the benefit of explicitly modelling the absence of terms. A naive Bayes classifier with a Bernoulli event model is not the same as a multinomial NB classifier with frequency counts truncated to one.

We have chosen to use *sklearn* machine learning API in Python for easy and fast prototyping. The model's best predictive accuracy we are getting 49%

b. Random Forest:

Our model with Naïve Bayes architecture could not achieve very high accuracy, therefore, we have decided to give Random Forest model a try. The main difference between Random Forest and Naïve Bayes classifier is that Random Forest is robust against overfitting, it gives better results with the increasing number of examples and it works good with numerical, categorical data where we need to estimate a distribution over continuous value for naive bayes.

Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random Forest is a supervised learning algorithm. It creates a forest and makes it somehow random. The "forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

We have used python *sklearn* machine learning library for the in-built Random Forest algorithm. We have started with 1000 decision trees and 42 random states. Then we fitted the training data in the model and compared the output values with actual class labels. The model's best predictive accuracy we are getting 61%.

IV. Results and Analysys

A. Accuracy Comparison:

We have used Python 3 in Anaconda build to build the model and testing it. We have executed the code for 20 times and took the best result in terms of accuracy on test data. The accuracy results for differet types of classifiers are shown in table1.

Model	Iterations	Best Predictive Accuracy
Gaussian Naïve Bayes	20	59%
Bernoulii Naïve Bayes	20	59%
Random Forest	20	61%

Table1: Accuracy Comparison of different classifiers

B. Misclassification Results

We have tested our model on 2000 test data and nearly 8500 train+validation data for both the LSTM and BiLSTM model and we found a slight improvement of accuracy on test data. We have fitted the model 20 times and predicted the emotion labels on the test data. We have also computed the confusion matrix on the final predicted result in comparison to the actual emotion labels on the original test dataset. We have calculated pointwise difference between the actual and predicted labels and plotted the differences as a graph. The results are reported below:

a. Confusion Matrix:

	ANGER	**FEAR**	**JOY**	**SADNESS**
ANGER	213	121	34	132
FEAR	70	258	35	138
JOY	37	88	265	111
SADNESS	71	128	54	245

Table 2: Confusion Matrix for test data using Gaussian Naïve Bayes

	ANGER	**FEAR**	**JOY**	**SADNESS**
ANGER	215	120	33	132
FEAR	70	258	35	138
JOY	37	88	265	111
SADNESS	71	130	52	245

Table 3: Confusion Matrix for test data using Bernoulii Naïve Bayes

	ANGER	**FEAR**	**JOY**	**SADNESS**
ANGER	192	162	77	69
FEAR	58	303	63	77
JOY	23	112	327	39
SADNESS	58	173	83	184

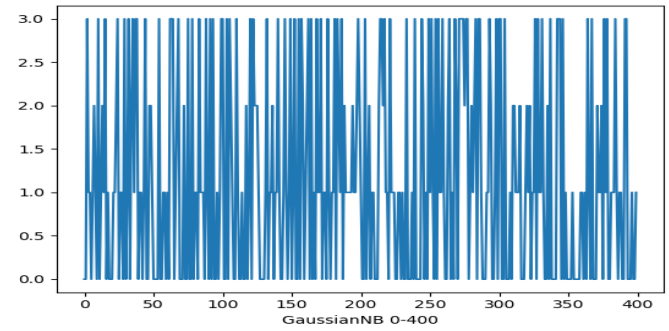
Table 4: Confusion Matrix for test data using Random Forest

A **confusion matrix** is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. The rows show the actual values and the columns show the predicted values for each of the classes. Table 2 shows the performance of the Gaussian Naïve Bayes model after 20 trial iterations for fitting the model with train+validation data and testing it with test data

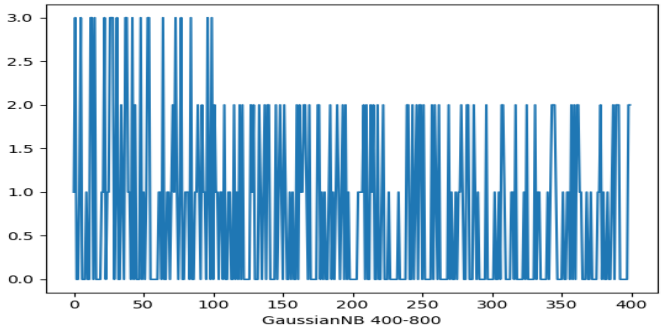
with test accuracy of 59%. Table 3 shows the same for Bernoulii Naïve Bayes model. Table 4 shows the performance of the Random Forest model after 20 trial iterations for fitting the model with train+validation data and testing it with test data with test accuracy of 61%. The diagonal elements represent the number of correctly classified examples and other elements specify the misclassifications. For example, 303 of total 500 *fear* test instances are correctly classified by our bidirectional LSTM model as *fear*, 58 instances are misclassified as *anger*, 63 instances are incorrectly classified as *joy* and 77 instances are incorrectly classified as *sadness* in the Random Forest confusion matrix. We can interpret all the values of the confusion matrices described in Table 2, and 4 by this example.

b. Plot of Variation:

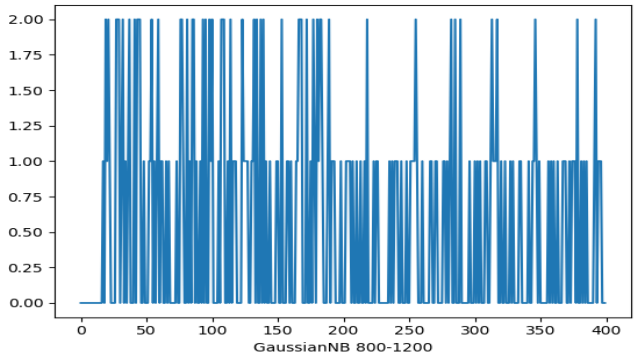
The actual labels for test data makes a one-dimensional vector of 2000 labels belonging to any of the *anger, fear, joy* or *sadness* classes. The predicted labels also produce a similar one-dimensional vector of length 2000. The elementwise difference of predicted test labels and actual test labels gives us a one-dimensional vector of length 2000. Each value v_i in variation vector is equal to $|predicted_i - actual_i|$ where $1 \leq i \leq 2000$. We have plotted the variation vector in graph. The number of our test instances are huge to fit the resultant graph in this paper in an interpretable manner. Therefore, we have divided our variation vector in five parts with $1/5^{th}$ of the total data in each part as follows: a. first 1-400 data points b. next 400-800 data points c. next 800-1200 data points, d. next 1200-1600 data points and e. next 1600-2000 data points and plotted them in graphs for both LSTM and biLSTM. The result of one model (LSTM) is shown in figure 3.



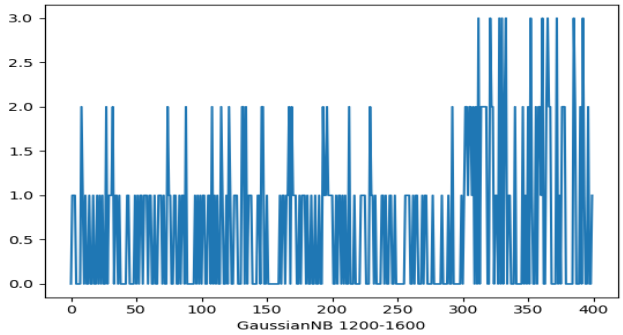
a.



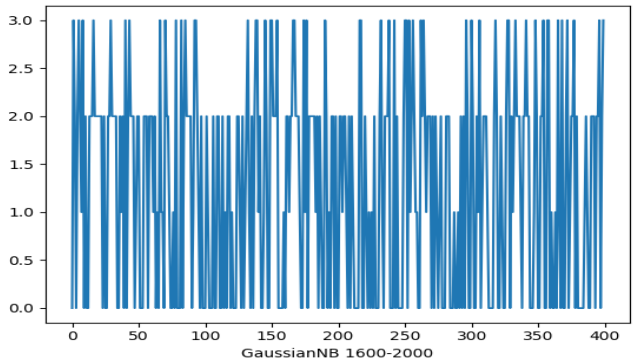
b.



c.



d.



e.

Figure 3: Elementwise difference of predicted and actual labels for Gaussian Naïve Bayes: a. first 0-400 points b. second 400-800 points c. third 800-1200 points d. fourth 1200-1600 points e. last 1600-2000 points.

c. Plot of Error Bar:

We have used Python in-build modules for creating models and they have in-build method for randomly initialization of weight matrices. Therefore, in each iteration, the weight initialization differs and the accuracy differs. We have plotted the variability of accuracy values in error bar graphs to indicate the *error* or variation in measurements of different iterations. Figure 4 shows the variation in the Test errors from 20 trial execution of the code. The mean, standard deviation and maximum and minimum range of values are indicated in the graphs.

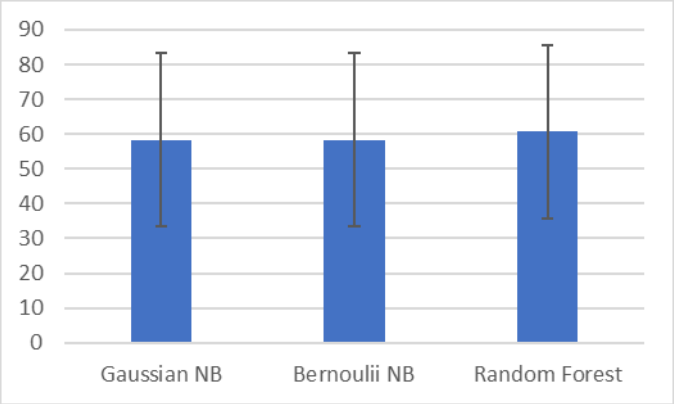


Figure 4: Error bar for Test error (20 executions)

The error bar with 95% confidence interval (i.e. a range of values of standard deviation which gives 95% certainty of containing the true mean of the values) is shown by the black lines in the blue bar chart.

V. Future Work and Conclusion

This project addressed the problem of identifying emotions in tweets. This is a specific kind of text mining problem. We could achieve best accuracy of 59% using single Naïve Bayes and 61% accuracy using Random Forest classifiers. We want to experiment with different kind of classifier models like AdaBoost and do some more hyperparameter tuning to increase the accuracy. Another way to accuracy improvement is to use some more semantic based feature selection methods and use the combined features as the corpus features. In the related work, we have seen that helps in improving accuracy by 10-20%. We also want to consider the Psycholinguistic which is being used recently to classify texts and sentences for a variety of tasks.

In future, we want to design an ensemble machine learning model to classify tweeter emotions as well as predict the emotion intensity of the classified tweets.

References

[1] S. Radha Krishna, R. Rajeswara Rao, "Automatic Text-Independent Emotion Recognition Using SpectralFeatures", in Journal of Innovation in Computer Science and Engineering,Vol. 5(1), July–Dec 2015@ ISSN 2278-0947.

[2] Taner Danisman, Adil Alpkocak, "Feeler: Emotion Classification of Text Using Vector Space Model".

[3] Swati D. Bhutekar, Prof. M. B. Chandak, "Corpus Based Emotion Extraction to implement prosody feature in Speech Synthesis Systems", in International Journal of Computer & Electronics Research, Volume 1, Issue 2,August 2012, ISSN : 2778-5795.

[4] Changqin Quan, Fuji Ren, "Sentence Emotion Analysis and Recognition Based on Emotion Words Using Ren-CECps", in International Journal of Advanced Intelligence Volume 2, Number 1, pp.105-117, July, 2010.

[5] Ali Houjeij, Layla Hamieh, Nader Mehdi, Hazem Hajj, "A Novel Approach for Emotion Classification based on Fusion of Text and Speech", in 19th International Conference on Telecommunications.

[6] Amira F. El Gohary, Torky I. Sultan, "A Computational Approach for Analyzing and Detecting Emotions in Arabic Text", International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622, Vol. 3, Issue 3,May-Jun 2013, pp.100-107.

[7] Shadi Shaheen, Wassim El-Hajj et.al, "Emotion Recognition from Text Based on Automatically Generated Rules",in 2014 IEEE International Conference on Data Mining Workshop.

[8] Parrott, W.G, "Emotions in Social Psychology," in Psychology Press, Philadelphia 2001.

[9] Jiawei Han and Micheline Kamber, "Data Mining Concepts and Techniques", Second Edition.

[10] https://sites.google.com/site/miningtwitter/questions/sentiment/

[11] http://www.tutorialspoint.com/mongodb/

[12] Hema Krishnan, M. Sudheep Elayidom, T.Santhanakrishnan, " Sentiment Analysis of tweets for inferring popularity of mobile phones", in International Journal of Computer Applications, Jan 2017 edition.

[13] Haowei Zhang, Jin Wang, Jixian Zhang, and Xuejie Zhang. 2017. YNU-HPCC at SemEval 2017 Task 4:Using a multi-channel CNN-LSTM model for sentiment classification. In Proceedings of the 11th International Workshop on Semantic Evaluation Vancouver, Canada, SemEval '17, pages 795–800.

[14] Jingjing Zhao, Yan Yang, and Bing Xu. 2017. MI&T Lab at SemEval-2017 Task 4: An integrated training method of word vector for sentiment classification. In Proceedings of the 11th International Workshop on Semantic Evaluation. Vancouver, Canada,SemEval '17, pages 688–692.

[15] Jeffrey Pennington, Richard Socher, Christopher D. Manning, Standford NLP group.GloVe: Global Vectors for Word Representation.