Emotion Detection from Text

Kopal Shinde
Student, Dept. Industrial & Manufacture Engineering
University of Michigan - Dearborn,
MI, USA
kopal@umich.edu

Modue Gaye
Student, Dept. Electrical & Computer Engineering
University of Michigan - Dearborn,
MI, USA
mlgaye@umich.edu

Reba Sahajadee
Student, Dept. Electrical & Computer Engineering
University of Michigan - Dearborn,
MI, USA
sreba@umich.edu

***Abstract*—The growing internet and social media has created a lot of data in form of text, video and voice. People are expressing their views and emotions with the help of different platforms available in the web of internet. With the growth of social media, emotion detection and classification has emerged as one of the important aspects in human interaction with machine and themselves. We have used sentences dataset from kaggle pre-labelled with emotions. These emotions are based on Plutchick wheel of emotions namely – joy, love, sadness, anger, fear and surprise. We are building and analyzing various emotion classification models and techniques to achieve greater accuracy in classification of emotion labels.**

Keywords—emotions, text, sentences SVM, Logistic Regression, KNN, Naïve Bayes, Random Forest Logistic Regression..

I. Introduction

Emotion Detection plays a promising role in the field of Artificial Intelligence, especially in the case of Human-Machine Interface development. For Emotion Detection from an artificial intelligence perspective different parameter should be taken into consideration. Various types of techniques are used to detect emotions from a human being like facial expressions, body movements, blood pressure, heart beat and textual information. This paper focuses on the emotion detection from textual information. Nowadays within the Internet there's an immense amount of textual data. It's fascinating to extract emotion from various goals like those of business. As some emotion expressions are culturally independent - for example, even in a foreign language where we do not understand the meaning of words it is relatively easy for anyone to recognize anger, scare, surprise, etc. in the message. Similarly in the expression of our face it is not so important if we grew up in Britain, USA or India, most of such expressions are very similar and have a similar meaning. The problems come with text documents. Any document or sentence is strongly dependent on the language it was written in. Even relatively similar languages have often different spelling and often also a bit different meaning and strength of some words. This is the reason why a model trained for one language is useless in another.

We were encouraged with the work done in the domain of sentiment analysis from text, speech and video for classifying into basic sentiments like positive, negative and neutral. We have gone through various research papers (included in the references) and got interested in classifying sentences with emotion labels. The research papers have discussed major challenges in sentiment analysis and ways to overcome them. They have covered different techniques to perform emotion classification and analysis.

We particularly found various amount of work done in twitter sentiment analysis and wanted to do more for improving the accuracy of emotion detection in text and exploring effects of different techniques for detecting the same. Bouazizi and Ohtsuki (2017) , few authors describe the effectiveness of classifying short text sequences (such as tweets) into anything more than three distinct classes (positive/negative/neutral). In particular, Bouazizi and Ohtsuki only achieved an overall accuracy of 56.9% or 60.2% on 5 distinct classes with only two classifiers (SVM and Naïve Bayes) in two of their papers. We are extending the existing knowledge of emotion detection to do classification using 7 different classifiers on 6 classes of emotions.

II. Methodology

We have proposed 10-step methodology to obtain a best model for training. Below are the steps involved:

a) Finding Labelled Emotion dataset
b) Creating Balanced and Unbalanced Dataset
c) Data Pre-Processing
d) Feature Extraction
e) Creating Train and Test sets
f) Training Model
g) Feature Selection
h) Cross Validation
i) Selecting Best Model
j) Classification

The Figure 1 below displays the proposed methodology with different steps and their sequence.

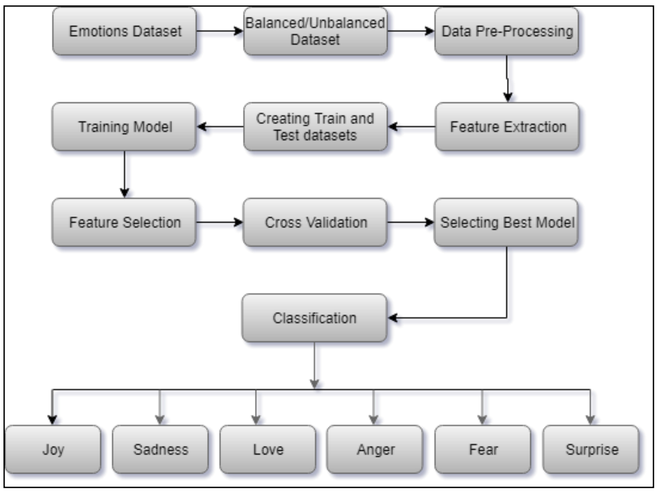

Figure 1: Proposed Methodology for Emotion Detection

Emotion analysis requires the use of standard machine learning algorithms like Logistic Regression, Naïve Bayes, Support Vector Machines (SVM), Radial basis SVM, Stochastic Gradient Classifier, KNN and Random Forest which are the basis of our classification design. These classifiers are trained using labelled data and hence fall under the category of supervised classification. Naïve Bayes and SVM are the traditional approaches which are diverse, but each has been shown to be effective in previous research papers for emotion detection studies. We extended the usual approaches by including more classifiers. This approach allowed us to see the behavior of different classification approaches and analyze how well they work for multi class problem. To implement these machine learning algorithms on the text emotion dataset, we followed the Data preprocessing steps, feature extraction, feature selection and different ML approaches. Below we will explain them in detail.

A. Naïve Bayes Classifier

The Naïve Bayes classifier is considered to be the simplest, trained, probabilistic classifier model. It is remarkably effective in several situations. We use a Multinomial Naïve Bayes model. With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial (p1….., pn) where pi is the probability that event i occurs (or K such multinomials in the multiclass case). A feature vector x = (x1……., xn) is then a histogram, with xi counting the number of times event i was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document (see bag of words assumption). The likelihood of observing a histogram x is

$$p(\mathbf{x} \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space:

$$\begin{aligned} \log p(C_k \mid \mathbf{x}) &\propto \log\left(p(C_k) \prod_{i=1}^{n} {p_{ki}}^{x_i}\right) \\ &= \log p(C_k) + \sum_{i=1}^{n} x_i \cdot \log p_{ki} \\ &= b + \mathbf{w}_k^\top \mathbf{x} \end{aligned}$$

In equation above, b=logp(Ck) and wki = logpki. The main shortcoming of the Naïve Bayes model is that it assumes each feature to be independent of all other features. This is the "naïve" assumption seen in the multiplication of P(f|c) in the definition of score. Thus, for example, if you had a feature best and another feature world's best, then their probabilities would be multiplied as though independent, even though the two are overlapping.

B. Support Vector Machine Classifier

Support Vector Machine is another popular classification algorithm. It is based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. SVM is primarily a classier method that performs classification tasks by constructing hyper planes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables. Radial Basis Function is a commonly used kernel in SVC:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right)$$

The computation of RBF kernel is comprised of the squared Euclidean distance and the exponential function.

I have used SVM with a linear kernel and radial basis kernel. For linear SVM, I used max_iter = 1000, penalty = L2. For radial basis kernel I used gamma value of 0.01 and C =1. The gamma parameter defines how far the influence of a single training example reaches, with low values. The C parameter trades off correct classification of training examples against maximization of the decision function’s margin.

C. Logistic Regression

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes. Logistic Regression can be of flowing types –

- Binary
- Multiclass
- Ordinal

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value. Below is an example logistic regression equation:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

In equation above y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in input data has an associated b coefficient (a constant real value) that must be learned from training data.

In logistic regression, I used multi_class = multinomial and solver = newton-cg. I applied this solver as it is used for multi-class classification problems and it is widely used.

D. Stochastic Gradient Descent (SGD) Classifier

SGD is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 10^5 training examples and more than 10^5 features.

Stochastic gradient descent is an optimization method for unconstrained optimization problems. In contrast to (batch) gradient descent, SGD approximates the true gradient of E(w,b) by considering a single training example at a time.

The class SGD Classifier implements a first-order SGD learning routine. The algorithm iterates over the training examples and for each example updates the model parameters according to the update rule given by

w←w−η(α∂R(w)∂w+∂L(wTxi+b,yi)∂w)

In equation above, η is the learning rate which controls the step-size in the parameter space. The intercept b is updated similarly but without regularization. The learning rate η can be either constant or gradually decaying. For classification, the default learning rate schedule (learning_rate='optimal') is given by

η (t)=1α(t0+t)

In equation above, t is the time step (there are a total of n_samples * n_iter time steps), t0 is determined based on a heuristic proposed by Léon Bottou such that the expected initial updates are comparable with the expected size of the weights (this assuming that the norm of the training samples is approx. 1). For regression the default learning rate schedule is inverse scaling (learning_rate='invscaling'), given by

η (t)=eta0tpower_t

In equation above, where eta0 and power_t are hyper parameters chosen by the user via eta0 and power_t, respectively. In SGDClassifier, I have used L2 penalty and loss = hinge as main parameters.

E. K-Nearest Neigbour

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

The k-nearest neighbor classifier can be viewed as assigning the k nearest neighbors a weight 1/k and all others 0 weight This can be generalised to weighted nearest neighbor classifiers. That is, where the ith nearest neighbor is assigned a weight. In KNNClassifier, I have used N neighbors as 5 as parameter.

F. Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates 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.

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set X = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B:

- Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
- Train a classification or regression tree fb on Xb, Yb.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

or by taking the majority vote in the case of classification trees.

This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

Additionally, an estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}$$

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the out-of-bag error: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.[13] The training and test error tend to level off after some number of trees have been fit. In RandomForestClassifier, we have used bootstrap = true and criterion = gini as the parameters.

G. Dataset

I have downloaded the Emotions dataset from Kaggle consisting of 105,000 observations and 6 classes. The content column consists of the sentences for which the emotions need to be detected. The emotions column consist of the labelled emotions for each sentence in content column. Below are 6 labels used in classifying the dataset –

- Joy
- Sadness
- Love
- Anger
- Fear
- Surprise

H. Balanced and Unbalanced Dataset

I picked 15,000 random observations to prepare unbalanced dataset from the emotions dataset of 105,000 observations. The observations picked for unbalanced dataset has uneven observations for 6 classes.

I picked 7,200 observations with 1,200 observations for each class to prepare balanced dataset from the emotions dataset of 105,000 observations. The observations picked for balanced dataset has equal observations for the 6 classes.

I. Data Preprocessing

The Emotion dataset is in the form of sentences which are having emotion labels. The sentences in the dataset need to be pre-processed before performing any type of operations in it. Before the dataset is passed to a learning algorithm, it needs to be pre-processed. The pre-processing is accomplished in steps mentioned below:

1. Converting all the content to lowercase.
2. Removing the URL's present in the sentences and replacing it with URL.
3. Removing @user from the content.
4. Removing words starting with #.
5. Perform spell check using textblob.
6. Remove the digits and special characters.
7. Remove the stop words using nltk.corpus for English words.

J. *Feature Extraction*

Feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations.

When the input data to an algorithm is too large to be processed and it is suspected to be redundant, then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

1) Stemming

Stemming is the process of replacing words (inflected words) to their root form or stems. For example, the words listener, listening, listened, etc are all reduced to their root word 'listen'. But the stemmed word may not be same as the morphological root of the word. For example, cookery is reduced to cookeri, which is not the actual word. In stemming, the ending characters of a word is just stripped to produce a stemmed form of the word rather than doing a dictionary look up to identify the actual root form of the word, which makes this technique computationally less expensive.
For this paper, we used Snowball stemmer (Porter 2), as it is the extended version of Porter Stemmer.

2) Lemmatization

Lemmatization is very similar to stemming, where we remove word affixes to get to the base form of a word. However, the base form in this case is known as the root word, but not the root stem. The difference being that the root word is always a lexicographically correct word (present in the dictionary), but the root stem may not be so. Thus, root word, also known as the lemma, will always be present in the dictionary.

3) Bag Of Words (BOW)

I am using BOW (Bag of Words) feature extractor as it is the simplest technique used for multi-text classification. It starts with a list of words called the vocabulary and then given an input text, it outputs a numerical vector which is simply the vector of word counts for each word of the vocabulary. Using BOW is assuming that more a word appears in a text, the more it is representative of its meaning. Therefore, we assume that given a set of text, a good classifier will be able to detect patterns in word distributions and learn to predict the emotion of a text based on which words occur and how many times they do.

To use BOW in python, CountVectorizer from scikit-learn library is used. It converts the text document collection into matrix of integers. This method helps to generate sparse matrix of the counts. The bag of words model ignores grammar and order of words.

4) Term Frequency – Inverse Document Frequency (TF-IDF)

It reflects the importance of a word in corpus or the collection. TF-IDF value increases with increase in frequency of a particular word in the document. In order to control the generality of more common words, the term frequency is offset by the frequency of words in corpus. Term frequency is the number of times a particular term appears in the text. Inverse document frequency measures the occurrence of any word in all documents.

TF-IDF score represents the relative importance of a term in the document and the entire corpus. TF-IDF score is composed by two terms: the first computes the normalized Term Frequency (TF), the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)

IDF(t) = log_e(Total number of documents / Number of documents with term t in it)

We used TF_IDF with stemmer and lemmatization for balanced and unbalanced data.

K. *Creating Train and Test Sets*

We have created train and test sets for both unbalanced and balanced emotions dataset. The X consist of the sentences column and the Y consists of labelled emotions. The dataset is split into training and test sets such that 30% of the data belongs to test set and 70% of data belongs to training set.

We have used SEED to get the same split every time for training and test set. The X_train represents emotional sentences and Y_train represents the emotion labels for training set. The X_test represents test set for emotional sentences and Y_test represents the emotion labels for test set.

L. *Training Model*

The trained data set is now passed to different models to get the best number of features and best n-gram to be used for every model. The graph in Figure 2 represents the best features and n-gram to be selected for SGD Classifier.

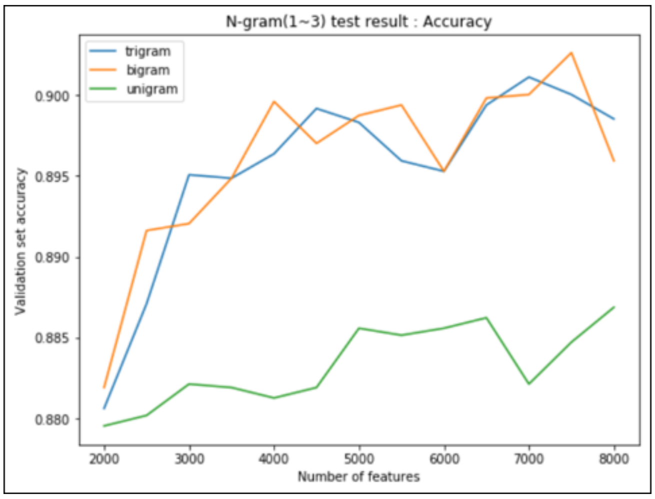

Figure 2: Graph showing accuracy at different # of features and n—grams for SGD Classifier

Similarly best number of features and n-grams have been selected for all models with combinations of balanced and unbalanced datasets and different feature extraction methods.

M. Cross Validation

All the models with best selected number of features and n-grams are split into 10-fold training and test sets to apply cross validation and obtain the average accuracy score. We This step is applied to each model with combinations of balanced and unbalanced dataset and different feature extraction methods.

Libraries used in python: Version 0.21 of scikitlearn libraries: TextBlob SnowballStemmer TfidfVectorizer MultinomialNB, SGDClassifier, LinearSVC, SVC, RandomForestClassifier, KNeighborsClassifier, accuracy_score, confusion_matrix, kFold, countectorizer and f1_score.

Libraries used in R: SnowballC, e107, rminer and the tm library.

III. Results

The results show that the highest Cross Validation accuracy of 90.09% is obtained for SGD Classifier for unbalanced dataset with feature extraction using Lemmatization and TF-IDF. The number of features to obtain the highest accuracy was selected as 7000 with Bi-gram as the n-gram.

Table IV represents analysis parameters for unbalanced dataset and a combination of lemmatize and TF-IDF as extraction method. Let us denote different algorithms with following alias names –

- A → Logistic Regression
- B → Multinomial Naïve Bayes
- C → Linear SVC
- D → Radial SVC
- E → SGD Classifier
- F → K-Neighbor Classifier
- G → Random Forest Classifier

Let us also denote the n-grams with following alias names –

- U – Unigram
- Bi – Bigram
- T – Trigram

TABLE I. Analysis of Different Algorithm Performance With Lemmatize and TF-IDF

	A	B	C	D	E	F	G
# Feat.	2500	2000	7000	7000	7000	6500	4000
n-gram	Bi	Bi	Bi	U	Bi	U	Bi
Acc.	87%	78%	90%	32%	90%	77%	86%
CV Acc.	87%	80%	90%	33%	90%	77%	87%
Prec.	0.87	0.81	0.9	0.1	0.9	0.77	0.87
Recall	0.87	0.78	0.9	0.32	0.9	0.77	0.86
F1-Sc.	0.86	0.77	0.89	0.15	0.9	0.76	0.86

IV. Discussion

In this paper, we have proposed a method to perform classification of sentences into six different emotion classes. We have implemented the classification using 7 different classifiers. All the classifiers were trained and tested with on the emotion sentences dataset. The analysis of the results obtained lead us to some conclusions about some features to be considered while classification. We concluded that there are many factors which determine how an algorithm performs. The type of dataset, if it is balanced or unbalanced is one of the factors. The other factor is the feature extraction method used like BOW or TF-IDF. Another major factor is determining n-gram and number of features to be selected for every algorithm used for text classification. The analysis of the results obtained lead us to some conclusions about some features to be considered while classification. We concluded that there are many factors which determine how an algorithm performs. The type of dataset, if it is balanced or unbalanced is one of the factors. The other factor is the feature extraction method used like BOW or TF-IDF. Another major factor is determining n-gram and number of features to be selected for every algorithm used for text classification.

A. Selecting Best Model

After cross validation is applied to all models, results for all models are analyzed. The parameters used for analysis are Model Accuracy, Cross Validation Accuracy, Precision, Recall and F1-Score.

1) Unbalanced dataset

Below are the tables for parameters analyzed on unbalanced dataset.

Table I represents analysis parameters for unbalanced dataset and a combination of stemmer and BOW as extraction method.

TABLE II. ANALYSIS OF DIFFERENT ALGORITHM PERFORMANCE WITH STEMMER AND BOW

	A	B	C	D	E	F	G
# of Feat.	5500	2500	7800	7800	6500	4500	6000
n-gram	Bi	Bi	T	T	Bi	Bi	T
Acc.	87%	83%	87%	87%	86%	65%	85%
CV Acc.	87%	84%	86%	76%	86%	66%	84%
Prec.	0.87	0.83	0.87	0.87	0.86	0.65	0.85
Recall	0.87	0.83	0.87	0.87	0.86	0.65	0.85
F1-Sc.	0.87	0.83	0.86	0.87	0.85	0.64	0.85

Table II represents analysis parameters for unbalanced dataset and a combination of stemmer and TF-IDF as extraction method.

TABLE III. ANALYSIS OF DIFFERENT ALGORITHM PERFORMANCE WITH STEMMER AND TF-IDF

	A	B	C	D	E	F	G
# Feat.	3000	2500	4000	7800	4500	5000	4000
n-gram	Bi	Bi	Bi	T	Bi	U	Bi
Acc.	85%	77%	88%	32%	85%	74%	84%
CV Acc.	85%	79%	87%	33%	88%	74%	84%
Prec.	0.85	0.81	0.88	0.1	0.88	0.74	0.84
Recall	0.85	0.77	0.88	0.32	0.88	0.74	0.84
F1-Sc.	0.84	0.75	0.87	0.15	0.88	0.74	0.84

Table III represents analysis parameters for unbalanced dataset and a combination of lemmatize and BOW as extraction method.

TABLE IV. ANALYSIS OF DIFFERENT ALGORITHM PERFORMANCE WITH LEMMATIZE AND BOW

	A	B	C	D	E	F	G
# Feat.	6000	2500	8000	5000	6000	8500	7500
n-gram	T	Bi	Bi	T	Bi	T	T
Acc.	89%	85%	89%	67%	88%	56%	87%
CV Acc.	89%	86%	89%	75%	88%	61%	87%
Prec.	0.89	0.85	0.89	0.75	0.88	0.6	0.87
Recall	0.89	0.85	0.89	0.67	0.87	0.57	0.87
F1-Sc.	0.89	0.85	0.89	0.62	0.87	0.56	0.87

Table IV represents analysis parameters for unbalanced dataset and a combination of lemmatize and TF-IDF as extraction method.

TABLE V. ANALYSIS OF DIFFERENT ALGORITHM PERFORMANCE WITH LEMMATIZE AND TF-IDF

	A	B	C	D	E	F	G
# Feat.	2500	2000	7000	7000	7000	6500	4000
n-gram	Bi	Bi	Bi	U	Bi	U	Bi
Acc.	87%	78%	90%	32%	90%	77%	86%
CV Acc.	87%	80%	90%	33%	90%	77%	87%
Prec.	0.87	0.81	0.9	0.1	0.9	0.77	0.87
Recall	0.87	0.78	0.9	0.32	0.9	0.77	0.86
F1-Sc.	0.86	0.77	0.89	0.15	0.9	0.76	0.86

2) Balanced dataset

Below are the tables for parameters analyzed on unbalanced dataset.

Table V represents analysis parameters for balanced dataset and a combination of stemmer and BOW as extraction method.

TABLE VI. ANALYSIS OF DIFFERENT ALGORITHM PERFORMANCE WITH STEMMER AND BOW

	A	B	C	D	E	F	G
# Feat.	4500	2500	5500	5500	5500	5000	2500
n-gram	T	Bi	T	Bi	T	Bi	Bi
Acc.	86%	80%	85%	71%	81%	59%	82%
CV Acc.	87%	81%	86%	76%	84%	58%	83%
Prec.	0.86	0.8	0.86	0.74	0.82	0.61	0.82
Recall	0.86	0.8	0.85	0.71	0.81	0.59	0.82
F1-Sc.	0.86	0.8	0.85	0.72	0.81	0.59	0.81

Table VI represents analysis parameters for balanced dataset and a combination of stemmer and TF-IDF as extraction method.

TABLE VII. ANALYSIS OF DIFFERENT ALGORITHM PERFORMANCE WITH STEMMER AND TF-IDF

	A	B	C	D	E	F	G
# Feat.	3500	3000	5000	3500	5500	5500	4500
n-gram	Bi	Bi	Bi	Bi	T	Bi	T

Acc.	85%	82%	86%	16%	86%	69%	80%
CV Acc.	87%	83%	88%	18%	88%	68%	83%
Prec.	0.85	0.82	0.86	0.19	0.86	0.69	0.8
Recall	0.85	0.82	0.86	0.16	0.86	0.69	0.8
F1-Sc.	0.85	0.82	0.86	0.04	0.86	0.69	0.8

Table VII represents analysis parameters for balanced dataset and a combination of lemmatize and BOW as extraction method.

TABLE VIII. ANALYSIS OF DIFFERENT ALGORITHM PERFORMANCE WITH LEMMATIZE AND BOW

	A	**B**	**C**	**D**	**E**	**F**	**G**
# Feat.	6000	6000	6000	4500	5500	6000	4000
n-gram	Bi	Bi	T	Bi	Bi	Bi	T
Acc.	88%	81%	88%	66%	86%	51%	85%
CV Acc.	89%	83%	89%	73%	87%	50%	87%
Prec.	0.89	0.81	0.88	0.7	0.86	0.57	0.86
Recall	0.88	0.81	0.88	0.66	0.86	0.51	0.85
F1-Sc.	0.88	0.81	0.88	0.66	0.86	0.51	0.85

Table VIII represents analysis parameters for balanced dataset and a combination of lemmatize and TF-IDF as extraction method.

TABLE IX. ANALYSIS OF DIFFERENT ALGORITHM PERFORMANCE WITH LEMMATIZE AND TF-IDF

	A	**B**	**C**	**D**	**E**	**F**	**G**
# Feat.	4000	4500	6000	4500	5500	3500	2500
n-gram	Bi	Bi	Bi	U	Bi	U	U
Acc.	88%	83%	90%	16%	89%	72%	84%
CV Acc.	89%	85%	90%	16%	90%	73%	85%
Prec.	0.88	0.83	0.9	0.02	0.89	0.72	0.84
Recall	0.88	0.83	0.9	0.16	0.89	0.72	0.84
F1-Sc.	0.88	0.83	0.89	0.04	0.89	0.72	0.83

Analysis of above Results:

1. KNN and redial basis SVC models performed poorly on both the balanced and unbalanced datasets.
2. LogisticRegression worked well with BOW and lemetization as well as stemmer.
3. LinearSVC performed well on both the datasets with TFIFD and Stemmer.
4. SGDClassifier performed the best with the combination of Stemmer and TFIDF. As well as combination of BOW and TFIDF.

V. CONCLUSION

Original hypothesis was to create Balanced and Unbalanced set and to get high accuracy for the balanced category based on the research paper. But results were opposite. The classifiers performed well on the unbalanced dataset than on balanced. But there was not much difference between the results. The reason is that our unbalanced category was not having high difference in observations with particular label, for example Anger will have 30% of observations and Joy will have 90%, in this case the joy classifier is more likely to dominate.

Based on previous approaches, the traditional approach worked well for SVM and Naïve Bayes but our classifier worked well with SGD classifier. It is because it depends on the dataset, it is not always the case that an algorithm which did great on one dataset may not work well with the other.

The BoW approach is a very simple approach to implement and understand but it is an order less document representation. BoW approach also cannot identify the semantics of the words but we can overcome this problem with word embedding approach.

VI. CONTRIBUTIONS OF EACH TEAM MEMBER

Kopal:

I worked through all the phases of project from problem analysis to solving major challenges and finally achieving good accuracy. Below are the main points of my work:

1. Finding different ways to clean a dataset for example trying different combinations, removing the missing words first before removing stop words and so on and finding how many features are left after applying the cleaning process.
2. Did analysis of why different cleaning approaches behave in different ways by doing experiments with the dataset. After applying the cleaning process, we got 10015 features for in the dataset.
3. I did not used the inbuilt function for feature selection rather I have written code for selecting the best features by looping over the features from minimum to maximum features.
4. I prepared a plot for choosing max features based on accuracy for different Ngrams (unigram, bigram and trigram) by using the function mentioned in point 3.
5. I have written a function to do classification report. This function is getting called for each classifier. This function will do 4 things in one go – a) It will do feature extraction, b) It will call the function mentioned in point 3 to perform feature selection and c) Doing cross validation and d) Finally printing the metrics (F1 score, Recall, Precision, Cross validation average accuracy).

Modou:

Find and downloaded different text emotion dataset. With the dataset consisting of 105,000 observations and 6 classes, I picked random observations to prepare balanced and unbalanced datasets.

A balance dataset has no bias hence it would allow for easier evaluation and results in more accuracy. This was a labelled dataset with sentences that needed preprocessing. The preprocessing was an enormous undertaking and was thus shared among us. I helped write the code to convert letters to lower cases and remove digits and characters. I also helped in the feature selection and the classification process which was easier.

Reba:

I duplicated the project in R.

1. I randomly selected 20,000 observations from the 40,000 tweeter dataset.
2. Explored the Bag of Words(BoW) and naïve bayes algorithm in R to create the model which has given us 32.98% of accuracy but the accuracy can be improved with parameter tuning, lemmatization and so on.

VII. Challanges

The major challenges were finding the best way for data cleaning and picking proper dataset. We initially tried to use a dataset text_emotion from Twitter, which was having 13 labelled emotions. We reduced the labels through the code based on their synonyms. We applied all the data cleaning steps on it and found that it was giving bad accuracy on all the classifiers around 36%, which was bad. We discussed with Professor Dave and he suggested to dig more on the cause and see if it works. We did lot of experiments:

1. Changed the sequence of cleaning process
2. Removing most frequent words
3. Removing least frequent words
4. Removing urls and usernames
5. Keeping the frequent words
6. Keeping the least frequent words
7. Applying all the classifiers each step1 through 3

After doing all the experiments, we observed that keeping least frequent words (1000) was significant as it contains the words that carries emotional weightage. Also, the most frequent words were important for the classification. So, we decided to keep both the least frequent and most frequent words. This way we improved the accuracy from 36% to 53% for twitter dataset.

But finally we dropped this dataset and applied all the work on other dataset that is mentioned in the dataset section. We dropped the twitter dataset because it was having more than 10000 observations labelled with Neutral class and we analyzed that it is Neutral class is not correct, it was having emotions that are falling in other different classes like Anger, Joy, Love. That's why we decided to drop this dataset and pick.

Another challenge was duplicating the work in R was figuring out the purpose of sparsity and reducing sparsity in textual data. After exploring on google about sparsity, it was clear that sparsity function cannot take values 0 or 1, only values in between which has solved the problem. We also had some issues implementing of KL divergence on the project. There wasn't enough time to write a wrote that would accurate calculate the difference in probability differences in the classifiers.

VIII. Future work

For future, the proposed system can used to test and predict on more datasets like customer reviews, twitter and social media comment, literature texts and scripts for movies and plays. It will be interesting to see how and which algorithm will perform better on other datasets. More classes can be used to classify the dataset with other emotion labels based on Plutchik's wheel of emotions. Also, there can be improvement made to detect sarcasm and emoticons or emoji.

If we would have more time for further investigation then, we might have tried to do unsupervised learning on amazon dataset. As it does not contain labels, we would have applied our model's learning to predict the customer comments on Amazon dataset and classify the six different emotions.

In R, we could've explored and implement other approaches like word embedding approach, Word2Vec woed embedding, Glove word embedding and fastText. We could try to find out the best performing classifier.

References

[1] Mondher Bouazizi, and Tomoaki Ohtsuki, "A Pattern-Based Approach for Multi-Class Sentiment Analysis in Twitter", August 2017.

[2] Cecilia Ovesdotter Alm, Dan Roth and Richard Sproat, "Emotions from Text: Machine Learning for Text-based Emotion Prediction.", January 2005.

[3] Uma Nagarsekar, Aditi Mhapsekar and Priyanka Kulkarni, "Emotion Detection from 'The SMS of the Internet'.", 2013.

[4] Radim Burget, Jan Karasek and Zedenek Smekal, "Recognition of emotions in 'Czech Newspaper Headlines'.", 2011.

[5] Hokuma Karimova, "The Emotion Wheel: What is It and How to Use it?.", December 2017.

[6] Tong Hui Kang and Chia Yew Ken "Multiclass Emotion Classification for Short Text" https://github.com/tlkh/text-emotion-classification

[7] scikit-learn, "Machine Learning in Python" https://scikit-learn.org/stable/index.html

[8] Sambit Mahapatra, "A Production ready Multi-Class Text Classifier.", March 2018 https://towardsdatascience.com/a-production-ready-multi-class-text-classifier-96490408757

[9] Saptashwa, "A Simple Example of Pipeline in Machine Learning with Scikit-learn.", November 2018. https://towardsdatascience.com/a-simple-example-of-pipeline-in-machine-learning-with-scikit-learn-e726ffbb6976

[10] Munezero, M. D., Montero, C. S., Sutinen, E., & Pajunen, J. (2014). Are they different? Affect, feeling, emotion, sentiment, and opinion detection in text. *IEEE transactions on affective computing*, *5*(2), 101-111.

[11] S.K. chinnamgari. *R Machine Learning Project*, Birmingham,UK, Packt, 2019,ch.4, pp. 115-129.