# Emotion Classification on Tweets

**Juilee Salunkhe, Sai Praharsha Devalla**
School of Information Studies, Syracuse University

**Dr. Bei Yu**
Syracuse University

## ABSTRACT

Micro blogging today has become a very popular communication tool among Internet users. Millions of users share opinions on different aspects of life every day. Therefore, micro blogging websites are rich sources of data for opinion mining and sentiment analysis. Because micro blogging has appeared relatively recently, there are a few research works that are devoted to this topic. In this paper, we are focusing on using Twitter, the most popular micro blogging platform, for the task of Emotion classification. This project proposes a method to classify text into four different Emotion-Categories: Sadness, Fear, Anger, and Joy. There are four unique emotions in this classification data. The approach is based on Machine Learning classification algorithms wherein we create a multi-label classifier and classify the tweets based on four categories.

***Keywords: Text-Mining, Linear SVM, Naïve Bayes and Multiclass Classification.***

## 1. INTRODUCTION

Emotions are described as intense feelings that are directed at something or someone in response to internal or external events having a significance for the individual. And the internet, today, has become a key medium through which people express their emotions, feelings, and opinions. Capturing these emotions in text, especially those posted or circulated on social media, can be a source of precious information, which can be used to study how different people react to different situations and events. Numerous marketers see success in their social media marketing strategies by paying closer attention to Twitter data analysis. Understanding which types of content and topics your audience members most enjoy can help drive your social marketing and content strategy. What is the point in sharing content no one cares about or enjoys? There are various applications of emotion detection in the text (tweets) which can be useful. The project intents to address the problem of detection, and classification of emotions of text in any form. We consider English text collected from social media like Twitter's tweets data. Social media like Twitter and Facebook is full of emotions, feelings, and opinions of people all over the world. Analyzing and classifying text based on emotions can be considered as an advanced form of Sentiment Analysis. This corpus of tweets was collected by polling the Twitter API for tweets that included emotion-related words such as anger, joy, Fear and Sadness.

## 2. DATA DESCRIPTION

The data set is obtained from the codalab website. The link to the data set can be found below this paragraph. This data set has two variables namely, 'ID', 'Tweet' and 'emotion'. The first variable contains the ID about the date on which the tweet was published, and the second variable contains the tweet of the corresponding ID. The 'emotion' variable is the target variable. The dataset is multi-classification with 4 different emotions. The given data set has a total of 8711 observations. From the graph below we can see that the data is slightly skewed.

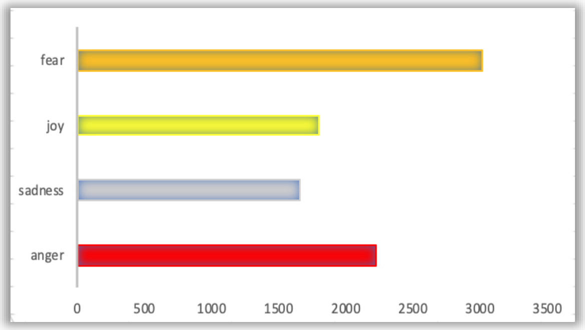


Fig:1 Distribution of label classes

## 3. METHODOLOGY

We shall discuss an unsupervised exploratory and 2 supervised text mining approaches that have been used for the project. The approach involves data preprocessing through tokenization (what to count) and vectorization (how to count) to convert the unstructured text data to structured data with a vector of numbers representing each text document. Once the text data has been vectorized and then transformed into a Term-Document Matrix, the required machine learning algorithm can be applied on the matrix to build models. The sklearn library has the required Countvectorizer [1] functions with all the required parameters suitable for converting a text corpus into a term document matrix. We use term document matrix to input LDA to get meaningful insights like we get to group similar tweets and give them a topic. For example, if we consider topic politics the tweets mostly be spread with anger and topic travel can be considered as joyous or happy. This is a hypothesis we try to show in this paper along with the classification of emotions based on the tweets. To classify the emotions two classification models linear SVM and Multinomial Naïve Bayes.

To evaluate multi-class classification models (Linear SVM and Naïve Bayes) we do use the accuracy, recall, precision and F1. For the balanced dataset accuracy can be used as an evaluation metric. But for an imbalanced dataset we should consider the macro recall, macro precision and macro F1 for evaluation of the model performance. We will use hold-out method to evaluate the model performance. As the data is imbalanced; accuracy, macro recall and macro F1 scores should be used. The model accuracy should be more than the majority vote baseline.

### 3.1 LDA Algorithm

LDA is a probabilistic model that tries to estimate the probability distributions for topics in documents and words in topics [2]. The LDA output provides us with a Topic level probability distribution of words which helps us to label the topics. The LDA output also consists of document level probability distribution of topics which helps us to assign a topic for each document. The number of topics can be tuned until we obtain coherent topics with minimum word intrusion [7]. Intruders are words which are out of place and do not belong together with the other words.

#### 3.1.1 Data Preprocessing

Started off with Tokenization to create a vocabulary (set of features) from the corpus. To reduce the size of the vocabulary the following filtering techniques were used:

i. *Removing Stop words*: Stopwords are unnecessary for a document, as they have no role in explaining the category of a document and are not at all helpful in text classification. Hence, they can be removed.

ii. *Minimum Document Frequency Filter:* We removed words that occur in very few documents by trying the min_df = 2 & min_df = 5 filter. This is done to improve the generalization performance and reduce overfitting.

iii. *Token Pattern:* Since the data is a tweet data there will lot of hashtags and references. So, I have considered the fact that every word should be of minimum length of 3 characters and only alphabets. This eliminates all the special characters which is noise for the model.

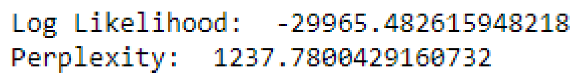
iv. *Maximum Document Frequency Filter:* We removed words that occurred in more than 80% of the documents, as they are most likely to be stopwords or other unimportant words which are not specific to any document.
v. We also applied the max_features = 2000 filter, to pick the top 2000 features that are picked based on term frequency across the corpus. Picking only the top 2000 features for Topic Modeling gave us better results, which we discuss in the below section.

After Tokenizing the next step was to vectorize the documents using the Term *Frequency* Vectorizer only, since the *TFIDF* vectorizer is not suitable for LDA algorithm. Since LDA is a probabilistic model, the IDF weighting of TF is not suitable for LDA. Hence, we used the Term Frequency Vectorization method.

The topic models were assessed using our human judgements to produce topics which are meaningful. The *coherence* of these topics was assessed using *word intrusion* and *topic intrusion* concepts. The assessment of these topic models will be discussed in the upcoming experimental results section.

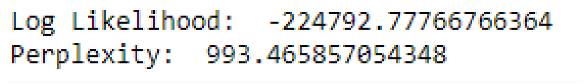
### 3.1.2 Experimental Results

Model 1: Initially, we started off with 10 topics and without the max_features, This model had 12010 features in the vocabulary, and it resulted in Log Likelihood & Perplexity scores as shown below:



```
Log Likelihood:  -29965.482615948218
Perplexity:  1237.7800429160732
```

Model 2: To improve the above performance, we modified the vectorizer by limiting the vocabulary to top 1000 features. This resulted in a better model performance with a lower word intrusion for the topics. A token pattern = [a-z][a-z][a-z]+. The Log Likelihood score increased, and the Perplexity score decreased as desired and the scores are as shown below:



```
Log Likelihood:  -224792.77766766364
Perplexity:  993.465857054348
```

If we see the words of few topics, we can relate them to emotion based on the category.

*Topic Musical concert: (amazing live) (live broadcast) (watch amazing) (lively musically) (nawaz sharif) (just got) (love new) (cheerfully accept) (love love) (awareness time) (just pleasing) (easy breezy) (breezy beautiful) (nick offerman) (long day) (like today) (guess just) (right great)*

If we look at the top tweets, the emotion they are carrying is joyous as it was related to musical event.

- When we give cheerfully and accept gratefully, everyone is blessed.
- Watch this amazing live.ly broadcast by @rosannahill.

*Topic Politics: (want) (lost) (just) (sadness) (week) (awful) (afraid) (thing) (dreadful) (away) (pout) (gbbo) (dark) (terrorism) (don) (pakistan) (time) (hope) (friend) (shake)*

If we look at the top tweets, the emotion they are carrying is fear and sadness as it was related to terrorism.

- Pakistan is the biggest victim of terrorism - Nawaz Sharif \nReally? It should have been biggest creator of terrorism.
- Very rare an officer just shoots without regard. They do not want that on their conscience.

The hypothesis which we made is true in a way based on the above two examples but needs to be done a lot of fact checking.

## 3.2 Multinomial Naïve Bayes:

A naïve Bayes classifier estimates the class-conditional probability by assuming that the attributes are conditionally independent, given the class label y [3]. The conditional independence assumption can be formally stated as follows:

$$P(X|Y = y) = \prod_{i=1}^{d} P(X_i|Y = y)$$

where each attribute set X = {X1, X2,..., Xd} consists of d attributes.

### 3.2.1 Data Preprocessing

The data is split into training and testing set where 70% of data goes to training and 30% for testing. We used two most used algorithms in text classification, Naïve Bayes, and Support Vector Machines. In case of Naïve Bayes, we used the Multinomial Naïve Bayes approach. The tweets are fed to the classification model were not lemmatized initially to check the performance. Then later lammatization is used on the tweets, the model performance has reduced. So, no lammatization method is used in the rest of the model building process. Stop words were removed and all the characters are converted to lowercase. We are using the count vectorizer with term frequency for the vectorization process. The minimum document frequency is set to 2 and ngram_range is tuned using Unigrams, Bigrams, Unigrams & Bigrams etc. were used. We used a token pattern to consider only words with alphabets [a-z][a-z][a-z]+. At the end, Hold-out test and cross-validation test results of best performing models are compared to check which method gives best results.

### 3.2.2 Experimental Results

From the below table it is evident that using n-grams is not a good idea for this data. The model with unigrams is performing well based on accuracy. Although the accuracy is decreasing with small range sometimes it makes sense to make bigrams to get the exact context. The Multinomial naïve Bayes model is classifying the tweets with an accuracy of 84%.

| | accuracy | precision | Recall | f1-score |
|---|---|---|---|---|
| unigrams | 0.841 | 0.842 | 0.841 | 0.841 |
| (uni+bi) grams | 0.838 | 0.840 | 0.838 | 0.837 |
| (uni+bi +tri) grams | 0.835 | 0.838 | 0.835 | 0.834 |

Table 1: Metrics of Multinomial NB

The top 10 features for anger and joyous classes from the multi nomial naïve Bayes best model are.

Top 10 anger words:

```
(-4.121765710437097,        'anger'),
(-4.039073994591983,         'rage'),
(-3.91045661676989,         'bitter'),
(-3.8498319949534547,       'fuming'),
(-3.8498319949534547,      'revenge'),
(-3.7628206179638255,     'offended'),
(-3.6162171437719497,      'outrage'),
(-3.4126181885307103,       'madden'),
(-3.4126181885307103,       'offend'),
(-3.4126181885307103,        'sting')
```

Top 10 joyous words:

```
(3.35357352612964,          'breezy'),
(3.3886648459409106,         'joyful'),
(3.3886648459409106,        'rejoice'),
(3.422566397616592,        'pleasing'),
(3.4553562204395822,          'cheer'),
(3.487104918754163,         'smiling'),
(3.734941082658744,       'musically'),
(3.759038634237804,            'glee'),
(3.759038634237804,          'lively'),
(4.0103530625187105,      'optimism')
```

## 3.3 Linear SVM

SVM is used to deal with high-dimensional data and avoids the dimensionality problem. It uses approach that represents decision boundary using a subset of the training examples, known as the support vectors. We use a concept maximal margin hyperplane, for linearly separable data linear SVM can be trained using this hyperplane. For linearly separable data the Model needs to find a hyperplane such it can place all the values of class 1 on one side and class 2 on other side of plane. The classifier will create infinitely many hyperplanes like that, now the classifier will need to choose one out of them which has the least training error to represent decision boundary. So that hyperplane performs well for the testing data [4]. A linear SVM is a classifier that searches for a hyperplane with the largest margin, which is why it is often known as a maximal margin classifier. The decision boundaries who have wide range tend to generalize the model than those with small margins. The model always looks for setting a large margin and low misclassification rate. Lower misclassification on training dataset does not mean lower misclassification on validation/testing data. To get a better a result of testing data, SVM looks for a higher margin. The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger margin separating hyperplane, even if that hyperplane misclassifies more points. For higher values of C, the margin will be less and misclassification will be less means the model has higher variance and the model tends to be overfitted. For small values of C, the model is more generalized as the misclassification will be more and model will have high bias. So, we need to find an optimal value to maintain the variance bias trade-off.

### 3.3.1 Data Preprocessing

We are following the same steps of Multinomial Naïve bayes for preprocessing the data. We are using the count vectorizer with term frequency for the vectorization process. The minimum document frequency is set to 2 and ngram_range is tuned using Unigrams, Bigrams, Unigrams & Bigrams etc. were used. We used a token pattern to consider only words with alphabets [a-z][a-z][a-z]+. At the end, Hold-out test and cross-validation test results of best performing models are compared to check which method gives best results.

### 3.3.2 Experimental results

From the above table it is evident that using n-grams is not a good idea for this data. The model with unigrams is performing well based on accuracy. Although the accuracy is decreasing with small range sometimes it makes sense to make bigrams to get the exact context. The Linear SVM model is classifying the tweets with an accuracy of 90%.

| | accuracy | precision | Recall | f1-score |
|---|---|---|---|---|
| unigrams | 0.884 | 0.885 | 0.884 | 0.884 |
| bigrams | 0.895 | 0.895 | 0.895 | 0.895 |
| trigrams | 0.893 | 0.893 | 0.893 | 0.893 |

Table 2: Metrics of Support Vector machine

From the above table it is evident that bigrams give the best results. The model with bigrams is performing well based on accuracy. The linear SVM performs better when compared with the multinomial naïve Bayes with same vectorization method and all parameters.

Now, for C value, which is penalty applied for misclassification on SVM model, I varied the value from 0.1 to 0.5 taking One-Hold out Accuracy Score into consideration. The table below shows the tuning of value of C to get the highest emotion Accuracy Score.

| C | Accuracy |
|---|---|
| 0.1 | 0.895 |
| 0.2 | 0.893 |
| 0.3 | 0.893 |
| 0.4 | 0.894 |
| 0.5 | 0.892 |

Table 3: Hyper Parameter tuning for C

The most important features for the best performing

Top 10 anger words

(0.8968690446349229, 'angry'),
(0.9062321576616679, 'offend'),
(0.9243026107808726, 'offended'),
(0.9268462738973147, 'furious'),
(0.9601926917265275, 'snap'),
(0.9702555501086053, 'madden'),
(0.9715170605001081, 'rage'),
(1.0472395348944064, 'bitter'),
(1.0545588952246616, 'revenge'),
(1.065822254594867, 'fuming')

Top 10 joyous words

(0.8695607449629719, 'smiling'),
(0.8814337434027376, 'breezy'),
(0.88379940342064, 'elated'),
(0.8885026481329541, 'lively'),
(0.9019140733061577, 'rejoicing'),
(0.9181140131095553, 'cheer'),
(0.9207691010252558, 'cheery'),
(0.9661417167794855, 'glee'),
(1.0251091529761611, 'optimism'),
(1.0301524191390283, 'hilarious')

### 3.4 Summarizing metrics of best models

From the below table the best model performances is obtained based on accuracy and other metrics we can say that Linear SVM using unigrams + bigrams as vectorization option.

| Model | ngrams | precision | Recall | accuracy |
|---|---|---|---|---|
| Multinomial NB | (1,1) | 0.842 | 0.841 | 0.841 |
| Linear SVM | (1,2) | 0.895 | 0.895 | 0.895 |

Table 4: Metrics of Support Vector machine.

Linear SVM is performing better with unigrams + bigrams as it takes context into account with little margin.

We have compared the results generated by hold-out method and cross-validation (6 fold) to verify that model is not overfitting.

| Model | Hold-Out | Cross-Validation (6 fold) |
|---|---|---|
| Multinomial NB | 0.841 | 0.851 |
| Linear SVM | 0.895 | 0.900 |

## 4. ERROR ANALYSIS

Anger is predicted as joyful.

Panpiper playing Big River outside The Bridges in SR1

@TheBarmyArmy looking forward to div 2 next year @AndyBarmyArmy? #leictershireaway #therey

@RiveraJose39 happy birthday to my seed!!!!!! #renewtherivalry #imyourdaddy #cantbeatme #tater

@KatzeAnilothei this amount of greedy mooching makes me snarl.

@CBSBigBrother never bring back Meech and Bridgette. Crying because someone looks at you? Ugh, and Bridgette

Most joyous tweets is predicted as anger.

#ukedchat A4 Just go outside (or to the gym hall) and play! \n #education #learning

@walterdonovanSS @mrb_rides_again evil, rich white men and their fucking cronies in intelligence/gov/academia using blithe blacks as fodder

Just watched Django Unchained, Other people may frown, but I titter in delight! 2/5

second day on the job and i already got a 45 dollar tip from a dude whose was constantly twitching his eye LOLOLOL

@Casper10666 I assure you there is no laughter, but increasing anger at the costs, and arrogance of Westminster.

Never make a #decision when you're #angry and never make a #promise when you're #happy. #wisewords

The models are unable to identify the sarcasm and negation effect. We tried using the ngrams to make the model understand the context and make predictions. But still the model is unable to understand the context to detect sarcasm and negation effect.

## 5. CONCLUSION AND FUTURE SCOPE

From the results we found linear SVM has the best performance with an accuracy score of 90% along with precision of 89% and recall of 90%. This shows that the model is performing well. There are no signs of overfitting as the model can generalize well.

There is scope to work on this project as the models are unable to classify the negation and sarcasm problem. Since the data is twitter data the words are not created properly. So, keeping that in mind we have to pre-process the data.

## REFERENCES

[1] [Online] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[2] [Online] https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2

[3] Tan PN, Steinbach M, Karpatne A, Kumar V (2013) Introduction to data mining, 2nd edition. Pearson, London.

[4] [Online] https://web.mit.edu/6.034/wwwbob/svm.pdf