CIS 520 Fall 2020, Project Report

Emotion Detection on Text by Machine Learning

Team Members:

- Zhigao Han; Email: `zhigao@seas.upenn.edu`
- Dinghong Li; Email: `dinghong@seas.upenn.edu`
- Chun Chang; Email: `chun3@seas.upenn.edu`

Abstract

The main work of this project is to extract human's feeling from text and classify. We take majority vote as baseline method for this project. We also try other machine learning methods like SVM, naive bayes, LSTM to make comparison with baseline method, and evaluate it. Finally, the LSTM model gets the best performance on the emotion detection test, which is 87.2% for test accuracy. Besides, we compare the result of other methods with LSTM model, and make analysis on it.

1 Motivation

Emotion Detection and Recognition from text is to detect and recognize types of human feelings through the expression of texts, such as anger, disgust, fear, happiness, sadness, and surprise. With the text data, which contains thousands of texts from social media and web page, we can train the model to find emotion behind text. This study is practical and easy doing, since it doesn't involve too much data cleaning. We will split text string into tokens and build model to classify it. This text analysis study can be applied to improve understanding of human's feeling. In real life practice, text analysis can help find customer's real feeling from feedback, which helps store or industry better adjust their marketing policy.

2 Related Work

Emotion Detection and Recognition from text is a recent field of machine learning . Emotion Analysis aims to detect and recognize human feelings through analysis on texts.Alexandra Balahur and his team deal with implicit affect in text using commonsense knowledge stored in EmotiNet, gathering and exploiting knowledge on emotion-triggering situations based on the Appraisal Theories[1]. Ali Yadollahi developed polarity classification for emotion-mining task[2]. His team introduced a set of important resources, including lexicons and datasets that researchers need for a polarity classification task. Kashfia Sailunaz studied emotion analysis[3] based on tweets dataset.The model introduces agreement score, sentiment score and emotion score of replies in influence score calculation.Bagus TrisAtmaja made SVM research on two stage dimensional emotion recognition by fusing predictions of acoustic and text network[4]. Experimental results show that this two-stage, late-fusion approach, obtains higher performance. BholaneSavita proposed sentiment analysis of twitter data using SentiStrength support vector machine (SVM). With SVM they increased the accuracy from 57.2%(twitter sentiment) to 62.3%[5]. Nazia Anjum Sharupa tried using naives bayes method to analyze emotion of tweets[6]. The average accuracy for different emotion category ranges from 60% to 80%. Study of using long-short term memory (LSTM)-based approach to text emotion recognition based on semantic word vector and emotional word vector of the input text[7] achieved a recognition accuracy of 70.66%, 5% higher than traditional CNN-based method.

3 Dataset

The dataset for project is Dataset from kaggle. There are over 13,000 instances in dataset(n=13000). Every instance has two parts: text and emotion label. Text is a short sentence (less than 30 words), which is the only feature (p=1). Emotion labels can be classified into six categories: sad, angry, love, surprise, fear, joy.The counts of each emotion category is shown in 1.

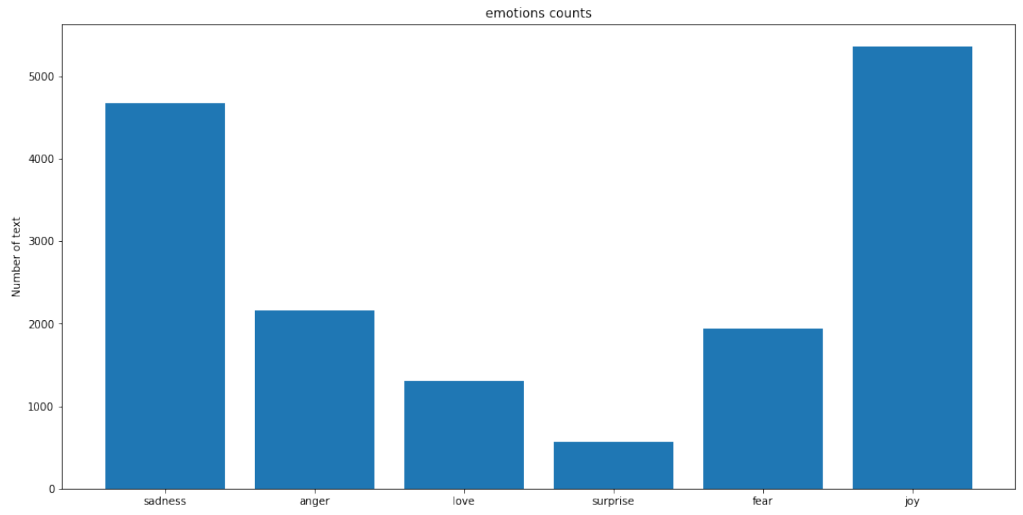
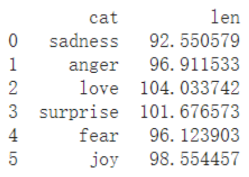


Figure 1: Counts of emotion categories

Sadness and joy have most counts, altogether take 80% . If we try horizontally move first three emotion categories : sadness, anger, love to last three positions. The emotion-frequency graph can treated as Gaussian distributed. Most common emotion that people have is neutral (no emotion). In other word, people's emotion swing between sadness and joy(happiness) in most time. Some emotion are rare, like fear. Since people generally do not display fear feeling through text, they are more likely to release their feeling by body language, like trembling. Also, we try analyze length of text for each emotion category. The result is shown below.



```
        cat         len
0   sadness   92.550579
1     anger   96.911533
2      love  104.033742
3  surprise  101.676573
4      fear   96.123903
5       joy   98.554457
```

Figure 2: Average length of text for emotion category

The length of text is distributed from 90 characters to 104 characters. The difference between highest text length and lowest text length is 1̃0 characters, less than two English words generally. In the following study, we can assume that text length for each emotion can be treated as a constant, which does not need further normalization. The Dataset from kaggle has been pre-processed by its provider in some way. He has removed all special characters, and replace all accented characters with an ascii counter-part. The data source only has alphabets, space and numbers. In our project, we still need some pre-processing on data. In our code we break text into tokens first, and use nltk.corpus to get rid of the stopwords in text. Also, some symbol and numeric character are cleaned. Using TF-IDF, we transfer a bunch of tokens into a vector of word weight. Then we have lemmazation and stemming on data to adjust weight, by reducing word to base form to prevent recalculation. Finally, we gain the data for experiment. The word cloud of most frequent words in cleeaned data is shown below.

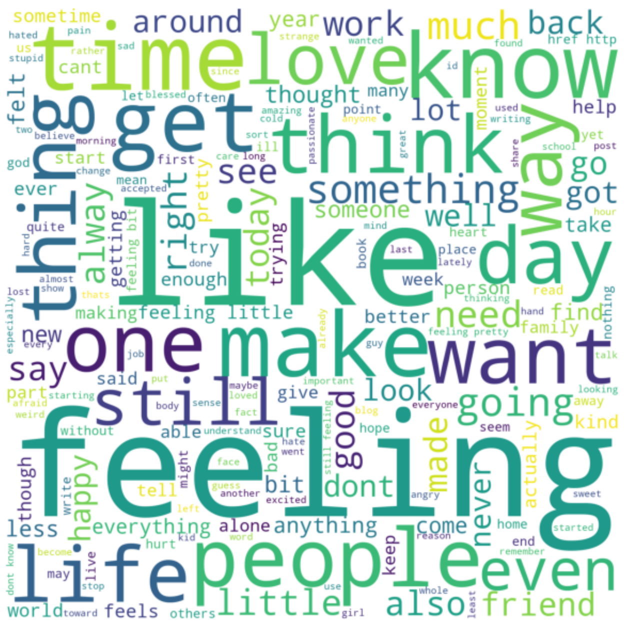


Figure 3: Wordcloud of most frequent words

4 Problem Formulation

The dataset we select for project has text and corresponding labels. Text can be treated actually series of words (ordered). In order to convert problem into machine learning algorithm analysis, we need to convert the text string into numerical feature vectors. In data preprocessing part, We used bag of words model. As a result, we divide each text file into tokens (splitting by space). Using TF-IDF, we transfer a bunch of tokens into a vector of word weight. Then we can apply this data into algorithm. We model the problem as a classification task. Each emotion label can be seen as a class. Under this setting, our goal becomes to classify a text into correct emotion category. We adopt cross entropy function as loss function. Cross entropy function calculates difference between two probability distribution. To optimize model, we have Adam SGD as optimization algorithm. We evaluate performance of our model by calculating its prediction accuracy on validation set and test set.

5 Methods

5.1 Baseline

We choose majority votes as the baseline model. That is, label all sentences with the label with the highest frequency. Majority vote iterate through the whole training document, text by text, to count the appearance of the word and its contribution the emotion. Then calculate the emotion score of the test document by summing up the contribution of each word, and the algorithm pick the largest one as the prediction. Apparently, this algorithm is very simple. We then try SVM and naive-Bayes models on the dataset.

5.2 SVM

As to SVM model, we calculate the TFIDF(the importance) of the word and unique identification number of the word. It projects each text until every text has same length and use TFIDF as its vectorized elements. We can thus do the classification in high dimensional spaces. The reason we adopt SVM is that SVM is a soft max model. There are some words in text might be ambiguous, like miss. For example, ”I miss you so much” can represent ”joy” emotion; while ”I miss the final” indicates a sadness emotion. The SVM model can minimize this kind dismatch case by hinge loss.

5.3 Naive Bayes

As for Naive Bayes model, we assume that the position of words do not matter and try use a whole bunch of words to predict the emotion category it belongs to. We calculate the frequencies of the word by using TDM (term-document matrix), to maximize the MAP of the probability of category.

5.4 LSTM

As to LSTM, we preprocess the input as SVM does. Our model will then feed the preprocessed data into two layers of LSTM to train the model. The structures of the neural network is listed in 5.4.

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 64)	320000
bidirectional (Bidirectional)	(None, 128)	66048
dense (Dense)	(None, 64)	8256
dense_1 (Dense)	(None, 7)	455

Table 1: structure of LSTM

5.5 Implementations

To implement method listed above, we will write a majority vote function, and mainly use sklearn library to apply other methods. We will implement a majority vote function and corresponding predict function. The majority vote function is to iterate text and update word counts and append emotion into training label. For SVM model, we apply the TFIDF processing function that we design to convert raw data into word vector. Padding each text until every text has same length and use TFIDF as its vectorized elements, then use sklearn.SVM to do the classification in high dimensional space. For naives bayes, we can directly ues sklearn.naive_bayes to fit our data and predict label. For LSTM method, we would use keras to directly relize it.

6 Experiments and Results

6.1 Experiment Settings

We evaluate performance of each model by calculating the prediction accuracy on the training set and the test set. Models we have experimented include majority votes, naive-Bayes, SVM and LSTM. We choose these models because they are widely used in NLP tasks.

Since we have modeled this problem as a classification task, it is appropriate to evaluate its performance simply by calculating its prediction accuracy on the test set.

SVM

penalty	kernal	test
1	linear	86.5
1	rbf	37.5
1	poly	38.3
2	linear	84.1
2	rbf	37.6
2	poly	38.5

Table 2: setting of SVM

LSTM

penalty	test
L_2	17.75
L_1	3.68
KL divergence	35.6
CrossEntropy	86

Table 3: setting of SVM

As for SVM, the method we chose turns the text into high dimensional space, and each text can be represented by a point in the that space. So it is separable. And since most of the word in the dictionary are not used for a single text, the distribution is sparse and the high degree of Kernal is not useful for this problem setting. The above arguments support the better performance of linear kernal.

As for LSTM, L2 and L1 loss are not usable because we are not looking for the minimum distance of the point in high space. Ex,$0, 1, 0$ has the same distance to $1, 0, 0$ or $0, 0, 1$. And the distance is diluted in the high dimensional space. Besides, the gradient vanishes after the ReLU layer.

So, we are using Cross Entropy, $H(p, q) = H(p) + D_{KL}(p \mid q)$ which maximize the difference between two distribution.

And for the architecture, LSTM is a one of recurrent neural networks. It can learn and remember/forget over long sequences of data. So we are using around 130(about a small paragraph) as the length/timestamp of the data.

6.2 Performance Results

As to our baseline model (e.g., majority votes), the prediction accuracy is 34.75% on the training set, and 41.5% on the test set.

model	training set	test
baseline	34.75	41.5
Naive-Bayes	79.738	73.350
SVM	93.888	86.350
LSTM	99.71875	87.2

Table 4: results

As to Naive Bayes model, the prediction accuracy is 79.738% on the training set, and 73.350% on the test set.
As to SVM model, the prediction accuracy is 93.888% on the training set, and 86.350% on the test set.
As we can see, all SVM models perform significantly better than the baseline model and the naive-Bayes model. It is because of its ability to generate information.
As to LSTM model, the prediction accuracy is 99.71875% on the training set, and 87.2% on the test set.
Compared with other algorithms, LSTM model can accomplish much better performance on the training set. It is because amazing ability to learn the distribution of neural networks. However, overfitting is also noticeable during this process. 4 clearly demonstrates this trend.

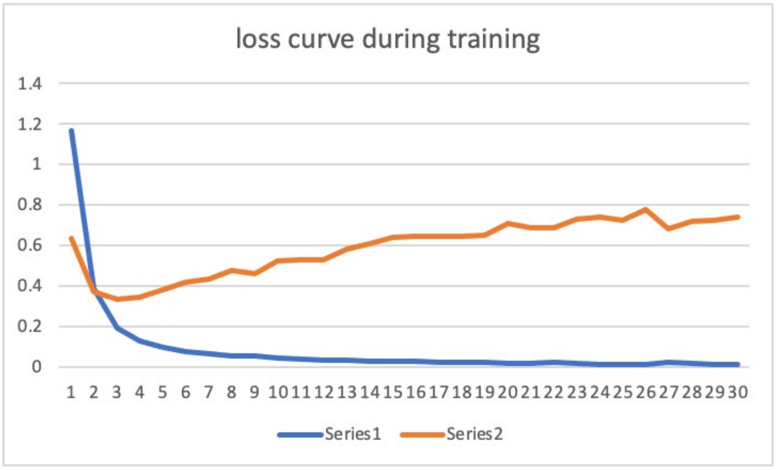


Figure 4: loss of LSTM on the training and validation sets

The major reason of this problem is that our training set is too small. It is also likely to be caused by our neural network is oversized.
6.2 summarizes the performance results across all of our models.

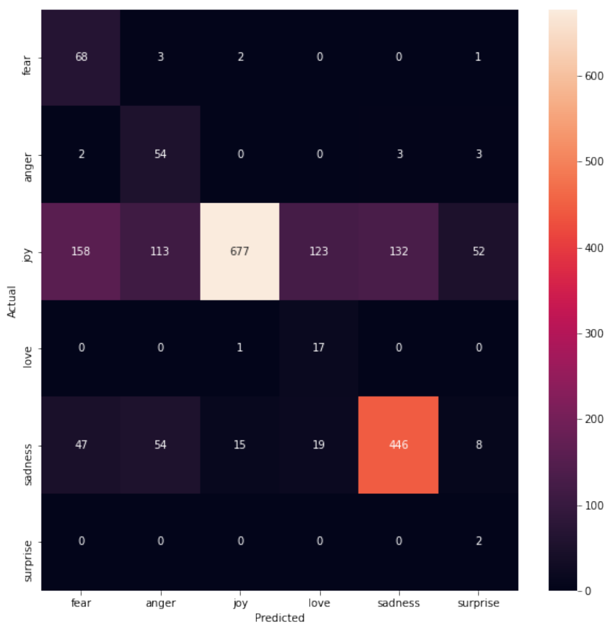


Figure 5: result of baseline model

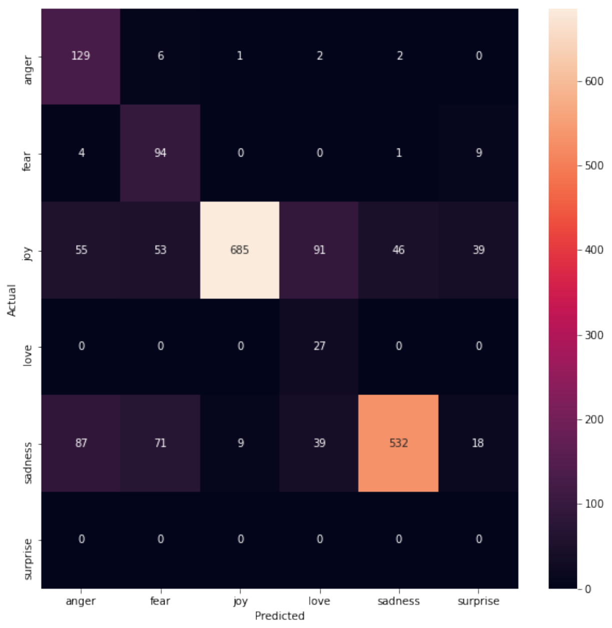


Figure 6: result of naive-bayes model

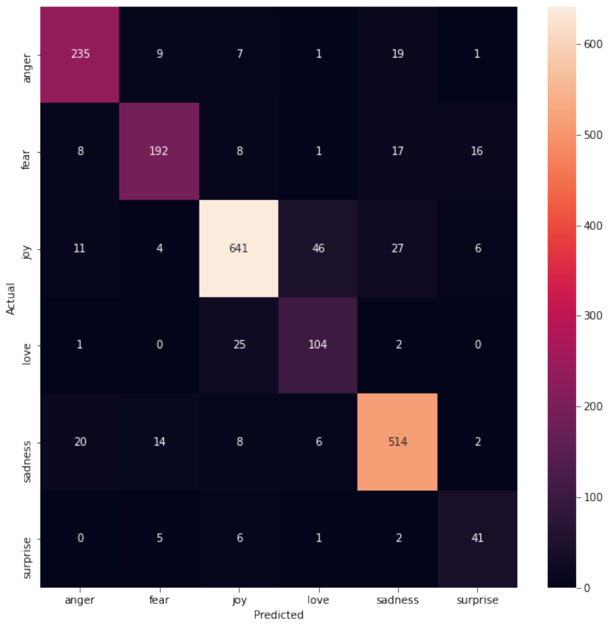


Figure 7: result of svm model

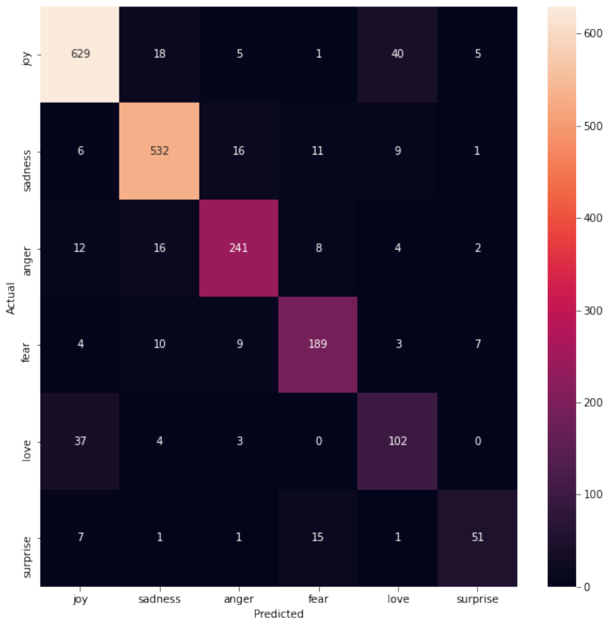


Figure 8: result of LSTM model

5 illustrates the prediction results of our baseline model. 6, 7 and 8 demonstrates the prediction results of Naive-Bayes, SVM and LSTM models, respectively.

7 Conclusion and Discussion

As we can see from the above result, the LSTM model can get the best performance on the emotion detection test. As to the test set, however, the advantage diminishes greatly.

It is because our LSTM model have a large number of parameters. This makes the model very fast to converge and fit the training data, but increases overfitting.

Naive Bayes works well because it assumes each word in the text is conditionally independent with other words, more specifically, it's statistically collecting bag of words and estimating the distribution of the text with the Bayes Rule. But it does not work perfect because the way the same words are used are different in terms of the context, and which is also the reason why "the bag of the words" has the accuracy cap around 70

Due to the nature of the our problem statement and dataset, the chosen methodology is supervised learning. The advantage of LSTM is LSTM could learn or forget selectively on the previous data. LSTM has large number of parameter to determine the "state" and "action" in each feature map, so it potentially learns the semantics under the vectorized text and characterizes the pattern that close words might have higher correlation/causality than the distant words do in the sentence.

The benefit of the SVM is the vectorized data with same catagory has similar distribution in high dimensional space. No matter by TFIDF or Tag Map, the human sentence follows specific pattern and shared the commonality. Therefore by utilizing the nonlinear feature feature of SVM, the emotion text can be better detected on high dimensional space.

As to future works, we can tune the parameters as well as initial conditions for the LSTM model to mitigate overfitting.

Acknowledgments

We are extremely grateful to the instructor of this class, Lyle Ungar. He gives up vivid class, so we have knowledge to finish this product. We also appreciate TAs in piazza community. They help us a lot when we post our confusion. We'd also say thanks to classmates in online class, their speech gives us inspiration for this project.

References

[1] Alexandra Balahur, Jesus M. Hermida, and Andres Montoyo. Detecting implicit expressions of emotion in text: A comparative analysis. *Decision Support Systems*, 53(4):742 – 753, 2012.

[2] Ali Yadollahi, Ameneh Gholipour Shahraki, and Osmar R. Zaiane. Current state of text sentiment analysis from opinion to emotion mining. *ACM Comput. Surv.*, 50(2), May 2017.

[3] Kashfia Sailunaz and Reda Alhajj. Emotion and sentiment analysis from twitter text. *Journal of Computational Science*, 36:101003, 2019.

[4] Bagus Tris Atmaja and Masato Akagi. Two-stage dimensional emotion recognition by fusing predictions of acoustic and text networks using svm. *Speech Communication*, 126:9 – 21, 2021.

[5] D BholaneSavita and Deipali Gore. Sentiment analysis on twitter data using support vector machine. *International Journal of Computer Science Trends and Technology (IJCST)–Volume*, 4:365, 2016.

[6] N. A. Sharupa, M. Rahman, N. Alvi, M. Raihan, A. Islam, and T. Raihan. Emotion detection of twitter post using multinomial naive bayes. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6, 2020.

[7] M. Su, C. Wu, K. Huang, and Q. Hong. Lstm-based text emotion recognition using semantic and emotional word vectors. In *2018 First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia)*, pages 1–6, 2018.