# Emotion Classification

Rico Pojol, Minhui Yu, Kelsey Fauntleroy, Shuang Du
COMP 562

May 5, 2022

## 1 Introduction

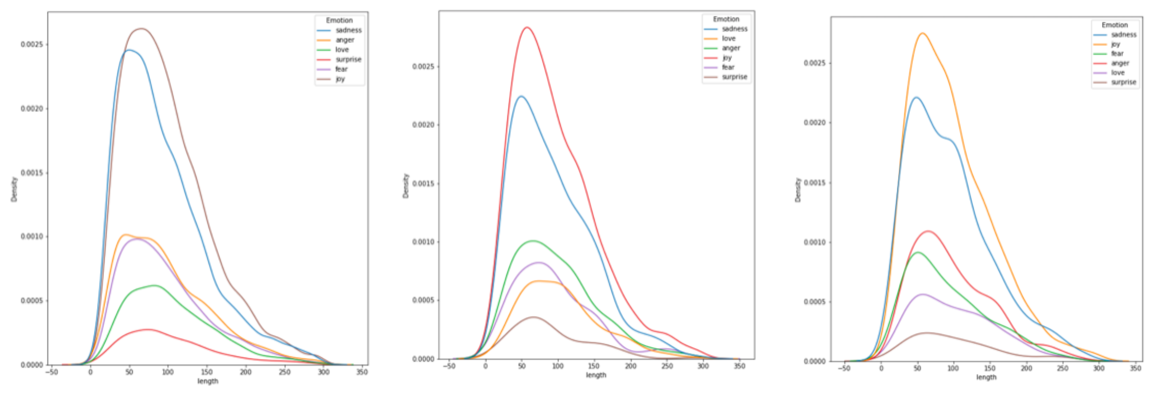
Text can be a conveyor of information, but classifying the emotion behind it can alter or enhance its perceived meaning. Being able to associate specific emotions with pieces of text such as social media posts, reviews of businesses, or customer service interactions allows a computer to interact with the human behind the text more appropriately and naturally. The goal of this project is to be able to classify the emotion behind a given text, so a computer or another person can take an appropriate action based on the resulting emotion from the classifier.

## 2 Approach

We used a dataset of pre-processed English Twitter posts [2], each already labeled with one of six emotions: sadness, anger, love, joy, fear, and surprise. The dataset was already split into test, training, and validation sets. To classify the dataset, we tried different classical supervised machine learning models as well as the BERT NLP model, and calculated the accuracy on the test set for each model.

## 3 Data Processing

Our dataset contains sentences which can be referred to different emotions - sadness, anger, love, surprise, fear and joy. The three separate files of data are loaded into three different variables, train_data, test_data and validation_data. In order to use this data, the text must be transformed into something our algorithm can digest.



(a) Train Data (b) Validation Data (c) Test Data

Figure 1: Data Distribution

To start the preprocess, label embedding is used to encode the emotion of each comment. A significant step in preprocessing is "cleaning" the data as much as possible in order to have an efficient model and algorithm. We achieved this by removing everything in the sentences that did not represent an emotion. Irrelevant characters and padding are removed. The process goes to the extent of removing "stop words", lowercasing every letter and using the porter stemmer algorithm for the stemming of each word. These sentences are then tokenized by converting them into lists. Every word of each sentence is encoded using one-hot embedding.

The clean sentences are then represented as the x-values for the three separate data variables (x_train, x_test, x_val). The encoding emotions for each sentenced that were embedded earlier are expressed as the y-values for the three separate data variables (y_train, y_test, y_val).

## 4 Methodologies and Results

### 4.1 Experiments with Classical Machine Learning Models

After transforming raw text into vectors using Bag of Words model (CountVectorizer), we have applied some classical machine learning models and compared their performance metrics.

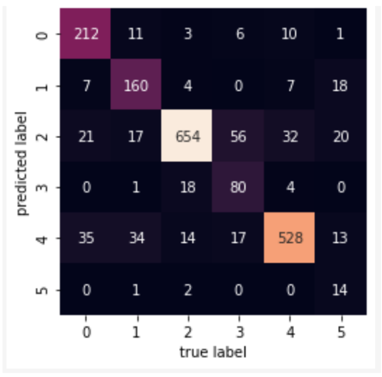
#### 4.1.1 Multinomial Naive Bayes Classifier

Multinomial Naive Bayes algorithm calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output. The advantage of Multinomial Naive Bayes Classifier is that it is easy to implement as we only have to calculate probability, and it is scalable for large datasets.

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial $(p_1, \ldots, p_n)$ where $p_i$ is the probability that event $i$ occurs. A feature vector $\mathbf{x} = (x_1, \ldots, x_n)$ is then a histogram, with $x_i$ counting the number of times event $i$ was observed in a particular instance. The likelihood of observing a histogram $x$ is given by

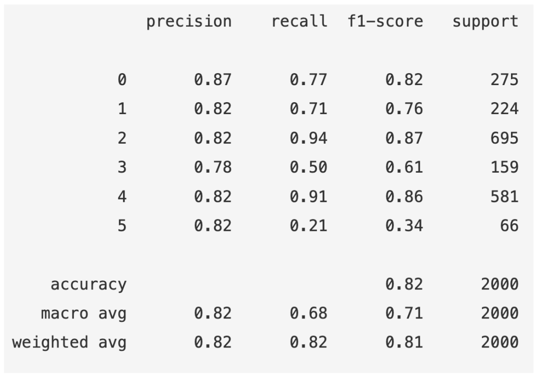
$$p(\mathbf{x} \mid C_k) = \frac{(\sum_{i=1}^{n} x_i)!}{\prod_{i=1}^{n} x_i!} \prod_{i=1}^{n} {p_{ki}}^{x_i} \tag{1}$$

Using the CountVectorizer transformer, we have removed the stop words, low-frequency words that appears less than 2 times in the corpus, and high-frequency words that appeared more than 80% times than others. We have 16000 observations in the training set, and 2000 observations in validation set and testing set separately. Applying the transformed corpus to the Multinomial Naive Bayes Model,the prediction accuracy reached 82%.



| predicted \ true | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 212 | 11 | 3 | 6 | 10 | 1 |
| 1 | 7 | 160 | 4 | 0 | 7 | 18 |
| 2 | 21 | 17 | 654 | 56 | 32 | 20 |
| 3 | 0 | 1 | 18 | 80 | 4 | 0 |
| 4 | 35 | 34 | 14 | 17 | 528 | 13 |
| 5 | 0 | 1 | 2 | 0 | 0 | 14 |

(a) Confusion Matrix



```
              precision    recall  f1-score   support

           0       0.87      0.77      0.82       275
           1       0.82      0.71      0.76       224
           2       0.82      0.94      0.87       695
           3       0.78      0.50      0.61       159
           4       0.82      0.91      0.86       581
           5       0.82      0.21      0.34        66

    accuracy                           0.82      2000
   macro avg       0.82      0.68      0.71      2000
weighted avg       0.82      0.82      0.81      2000
```

(b) Classification Report

Figure 2: Naive Bayes Classifier

#### 4.1.2 Comparison between different models

Besides applying Multinomial Naive Bayes model, we also applied several different classifiers, including Multinomial Logistic Regression, KNN, CART and SVM classifier to the corpus.

Below are some brief introduction of these classifiers:

1. Multinomial Logistic Regression is used when the dependent variable in question is nominal. It is commonly used as an alternative to naive Bayes classifiers because they do not assume statistical independence of the random variables (commonly known as features) that serve as predictors.

2. KNN (K Nearest Neighbors) is one of many (supervised learning) algorithms used in data mining and machine learning, it's a classifier algorithm where the learning is based "how

similar" is a data (a vector) from others.

3. CART (Classification And Regression Trees) is a is a classification algorithm for building a decision tree based on Gini's impurity index as splitting criterion. CART is a binary tree built by splitting nodes into two child nodes repeatedly.

4. SVM (Support Vector Machine) uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs

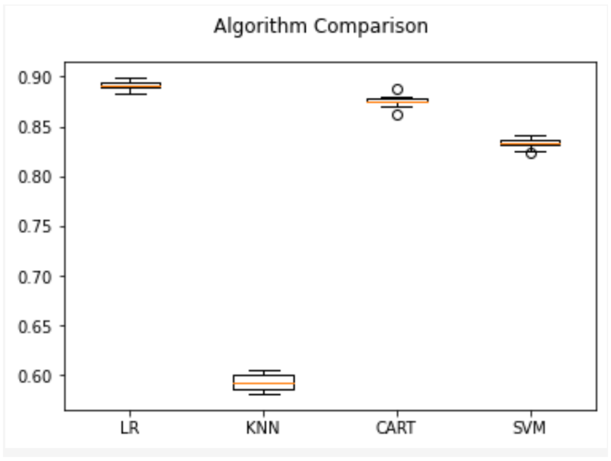


Figure 3: Algorithm Comparison

## 4.2 Experiments with State-of-the-art NLP Models

To do a comprehensive exploration of potential methodologies to solve out task, we also deployed the state-of-the-art NLP model Bidirectional Encoder Representations from Transformers (BERT) to compare with the classical machine learning approaches for our task.

### 4.2.1 BERT

BERT is a model built based on Transformer's encoder, stacked with the encoding structures containing the multi-head self-attention layers, normalization layers, and the feed forward neural network [1, 3]. It uses a bidirectional way to encode the input based on the self-attention mechanism, which means any word can 'communicate' with all words of the sentence to obtain a best deserved attention score. For the application, BERT is designed to be pretrained on very large corpus in a self-supervised manner, and then fine-tuned on task-specific datasets. During pretraining it aims to predict the randomly masked words in a sentence, as well as predict whether a sentence is the next to another sentence. With this multitask way of training, the model has great context awareness comprehensively and can be used for a wide variety of tasks. There are tens of thousands of pretrained BERT model in hugging face [1] aiming for varieties of tasks and languages.

### 4.2.2 Training details

Before being input to the model, the data was tokenized and encoded in following steps. Firstly, each word and character of the sentences were separated, and [CLS] and [SEP] tokens were added to the sentences to mark the beginning and ending of the first sentence. Then, all the tokens were convert into integer input IDs, token type IDs and masks based on the BERT tokenizer. We used the max sentence length of 50 in the experiments, so the sentence shorter than 50 words were filled with paddings until it reach 50 words, and the words beyond 50 were discarded. Then we mask the padding with 0, words with 1, and convert the data and masks into tensor to enable the model to load them.

In our case, for both data encoding and training, we used the "bert-base-cased" model, which was pretrained in case-sensitive English. We added a dropout layer and a linear layer with relu as the task layers. The batch size used was 32 and the optimizer used was Adam. The model was trained 12 epochs to have the best accuracy, and would start to over-fit afterwards based on out experiments. The final accuracy on the test set is **0.913±0.001**.

[1]https://huggingface.co/models

Table 1: Result comparison between the classical and SOTA NLP methods in the form of mean±standard deviation (best results are shown in boldface).

| Method | ACC |
|---|---|
| LR | $0.89 \pm 0.005$ |
| KNN | $0.59 \pm 0.008$ |
| CART | $0.88 \pm 0.006$ |
| SVM | $0.83 \pm 0.005$ |
| BERT | $0.913 \pm 0.001$ |

## 5 Discussion

Comparing classical machine learning models with the state-of-the-art BERT NLP model, it is clear that a model tailored for NLP like BERT will result in a higher accuracy when training a model. However, as our problem was not very complex, the difference in accuracy between the BERT model and the classical supervised learning models was in general not very large.

### 5.1 Classical Models

As is shown in table 1, between the classical models, we reported accuracies of **0.82** for our Multinomial Naive Bayes model, **0.89** for our Multinomial Logistic Regression model, **0.59** for a K-Nearest Neighbors model, **0.88** for a Classification and Regression Trees model, and **0.83** for a Support Vector Machine model. While most of our models reported an accuracy between 0.80 and 0.90, the K-Nearest Neighbors model reported a significantly lower accuracy, meaning that data points of different classes may not have very clear separation between each other when their features are interpreted as Euclidean coordinates. Such consistent results from the other classical models suggests that for a non-complex text classification problem, basic models may be sufficient.

### 5.2 BERT

The BERT model resulted in a higher accuracy of $\mathbf{0.91 \pm 0.001}$. We suspect the main reason the BERT model performed with higher accuracy compared to the other models is that the BERT model uses an encoder that processed the text in the corpus in a more appropriate manner for NLP classification. The BERT model is also pre-trained for NLP tasks, which contributes to its higher accuracy.

### 5.3 Conclusion

From training our classical supervised learning models, we found that even basic machine learning models can be accurate tools for classifying the underlying emotion of texts, with even greater accuracy found in an NLP specific, more advanced model. This means companies and businesses may be able to employ fairly lightweight ML approaches to enhance user experiences based on feedback that customers or reviewers may input. However, processing text alone does not account for nuances in spoken language that are not covered in our corpus. Furthermore, our dataset was originally sourced from Twitter posts, which may only represent a specific subset of kinds of text found online. Further steps in this project would include using a corpus of longer or more complex texts, or texts with nuances from speech that may introduce further emotion than what text alone conveys. However, we expect that the classical machine learning models we employed in the project may not be sufficient when applied a more complex corpus, and that more specialized models such as BERT would need to be utilized.

## References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.