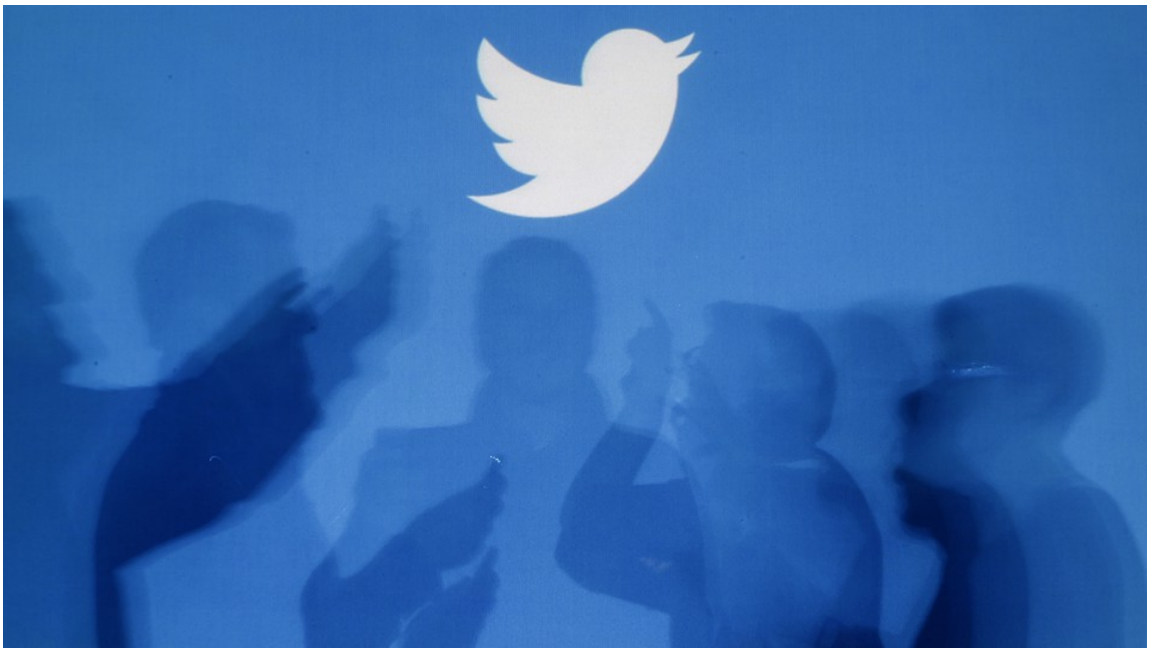
Sentiment Classification using Language Modeling

JUN HWEE OH, University of California San Diego
LINDA LIN, University of California San Diego
HENRY CHAN, University of California San Diego



0 ABSTRACT

With the advancement of technology and the rise of a wide range of data available today. The Internet has become the most common way for humans to gather and exchange information. Social networks like Twitter and TikTok often become the first new source of information as well as the platform to express and discuss feelings. We are going to focus on the tweets that the person has and analyze the mood behind the tweet using Natural Language Toolkit and widely available machine learning techniques.

ACM Reference Format:
Jun Hwee Oh, Linda Lin, and Henry Chan. 2022. Sentiment Classification using Language Modeling. 1, 1 (March 2022), 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: Jun Hwee Oh, j6oh@ucsd.edu, University of California San Diego; Linda Lin, jul052@ucsd.edu, University of California San Diego; Henry Chan, chchan@ucsd.edu, University of California San Diego.

Manuscript submitted to ACM

1 INTRODUCTION

With the advancement of technology and the rise of a wide range of data available today. The Internet has become the most common way for humans to gather and exchange information. Social networks like Twitter and TikTok often become the first new source of information as well as the platform to express and discuss feelings. The data behind the social network has become the most valuable way to extract information about that specific person and the most valuable information of identifying the person's interest. Together with other data like location and time. Companies can use this information to push more personal specific and relative information to that person.

In this paper, we are going to mainly focus on the tweets that the person has to analyze to analyze the mood behind the tweet using widely available machine learning techniques that are available today as well as sentiment analysis.

Sentiment analysis is a natural language processing (NLP) technique used to determine where data is positive, negative, or neutral. It is very useful to analyze massive amounts of data automatically. We also use various machine learning algorithms like Naive Bayes, Support Vector Machine together with sentiment analysis to have a general understanding of our data.

2 DATASET

In this project, we are going to implement a Twitter sentiment analysis model that helps to identify the sentiment of the tweet. The data set provided is the Sentiment140Dataset.csv, which consists of 1,600,000 tweets that have been extracted using the Twitter API. There are six columns present in the data set

- target: the rating of the tweet (0 is negative and 4 is positive)
- ids: unique id of the tweet
- date: date of the tweet
- flag: refers to the query. If no such query exists then it is NO QUERY
- user: name of the user that tweeted
- text: the actual content of the tweet

We are using python pandas to do the data processing. Pandas is an open-source Python package that is most widely used for data science/data analysis and machine tasks. We mostly use it to manipulate data.

Matplotlib, seaborn, and ploty are the most common packages to visualize and interact with the data. We use it together with pandas to present graphs and results of our project.

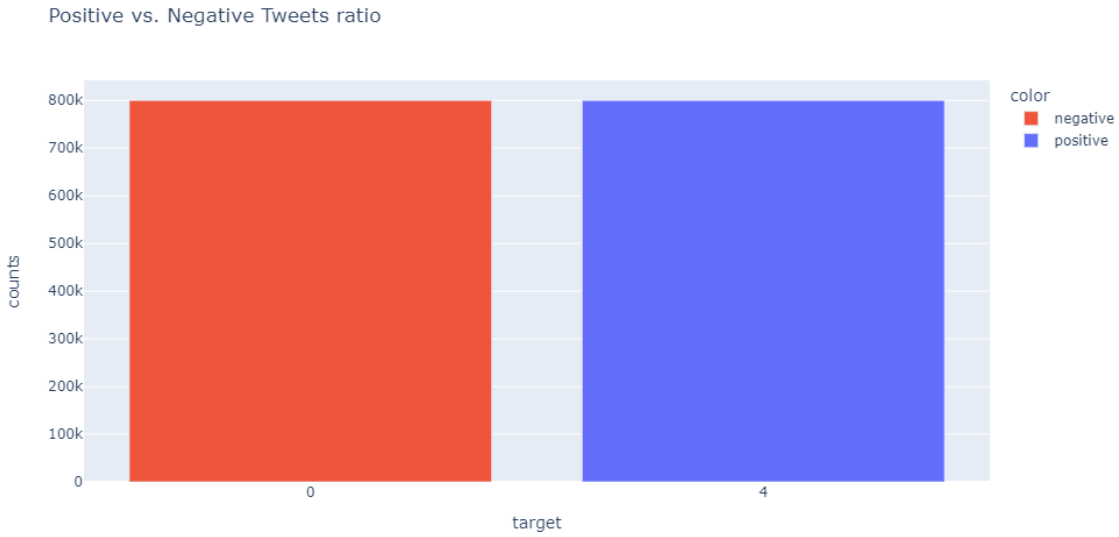
Numpy is a python package specifically designed for fast matrix calculation. It is the algorithm behind many machine learning models we choose.

Sklearn, XGBoost, and BERT are an example of machine learning library design. It will be our main source of machine learning models to choose from.

Natural Language Toolkit (NLTK) is the most common package used for language processing. It is the course for turning our data into machine learning compatible.

3 PREDICTIVE TASK

Before we start doing any machine learning, it is a great idea to know the data we are working with. As we do a basic exploratory data analysis, we can immediately find out that the dataset has been selected so that there are an equal amount of positive and negative tweets (800000 tweets in each category) and no neutral tweets. We also find out that the data does not contain any missing value.



We also quickly find out that the tweets contain non-ASCII characters. Luckily, the dataset also has been selected so that no logogram, syllabary, abjad, and abugida language is involved. The text is mostly alphabetic characters

Based on the EDA we analyze, we can follow the scikit-learn algorithm cheat-sheets to select our algorithm. Our prediction will be categorical(positive or negative). We choose the classification model to build our model. We also try popular alternative machine learning algorithms like XGBoost and BERT to have another view of our result.

4 DATA CLEANING

Although the dataset does not contain missing data. It still contains a lot of information that may skew our machine learning model. A tweet can contain opinions, expressions, or even a conversion or any of the combinations above. The result of the data will consist mostly of words that will contribute to the ratio of positive and negative in a sentence. The raw tweet can contain the following not useful information and we will try to remove it:

- Transform all the capital letters into lowercase
- Expend all the contraction (e.g. can't -> cannot, I'll -> I will)
- Remove URLs (e.g. www.google.com)
- Remove hashtags (#hashtags) and mentions (@user_name)
- Remove all punctuation, symbols, and numbers
- Remove stopword
- Stemming of our word (studies -> study, running -> run)

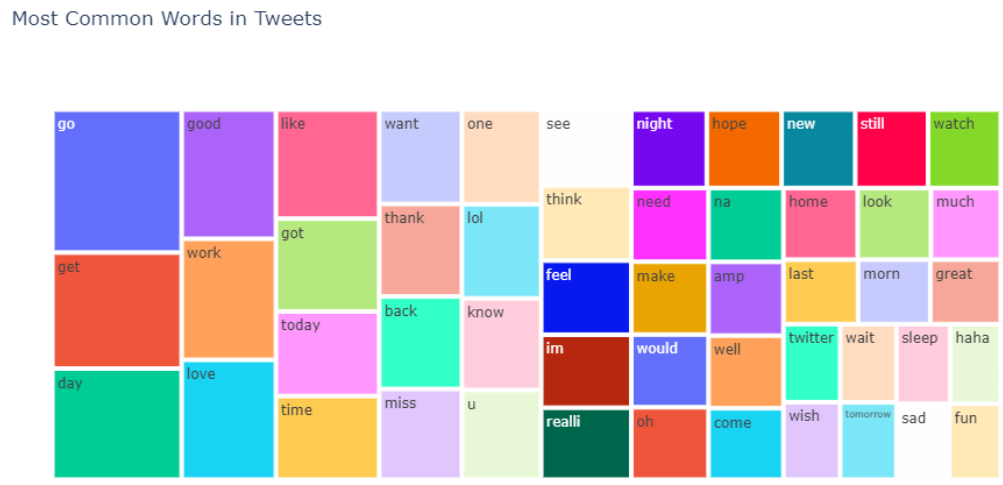
Below are some of the key points of cleaning:

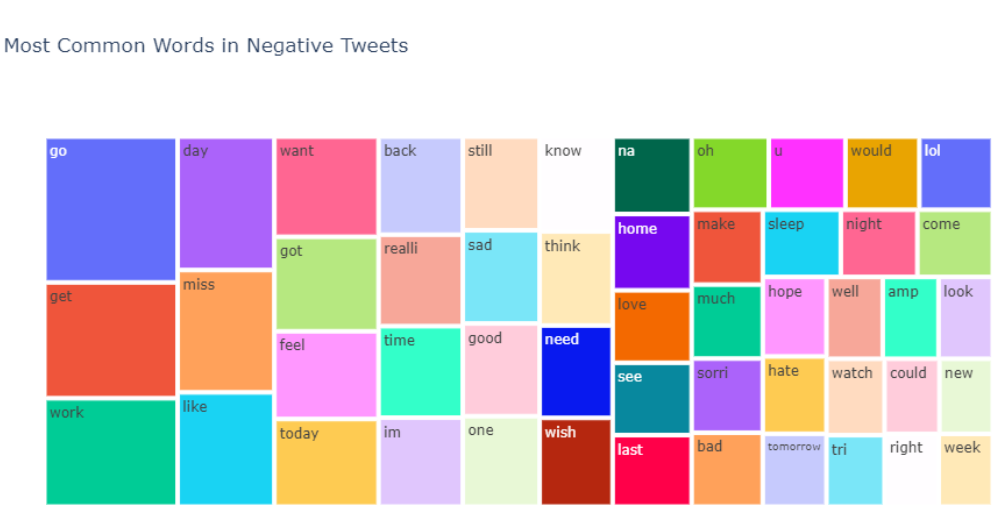
- Transform all the capital letters into lowercase, we apply the python function 'str.tolower()'
- To remove contraction, we build a simple mapping of some of the most frequently used contractions like 't -> not, 's -> is, and expand it.
- To remove mentions, we build a regular expression pattern expression to remove the words after the @user_name

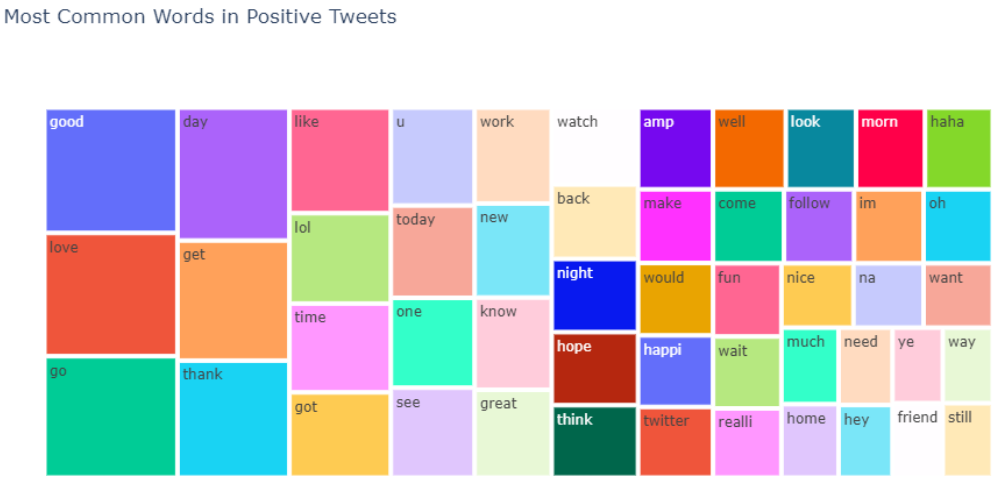
- To remove hashtags, we build a regular expression pattern to remove the word after the #hashtags
- To remove repeating characters, we build a regular expression pattern that identifies any repeating character and reduce it to only a single character. We choose three repetition words as the minimum because in English it is normal to have two repeating words.
- To remove URLs, we build a regular expression pattern to identify the keywords like 'HTTP' 'HTTPS' and 'WWW' and remove the following after it.
- To remove stop words, we used the NLTK library to help us identify the English stopping word. The NLTK provides a compiled list of stopwords that are presented in English.
- To remove punctuation, symbol, and number, we a regular expression pattern and remove it
- Finally, we tokenize the data and stem the word using the NLTK library.

5 EXPLORATORY DATA ANALYSIS

As we finished cleaning out data. We also count the most frequent words that are used in tweets that have been identified as positive and negative and combined. The results are very similar, but it does include some positive/negative specific words that are only present in positive/negative sentences.

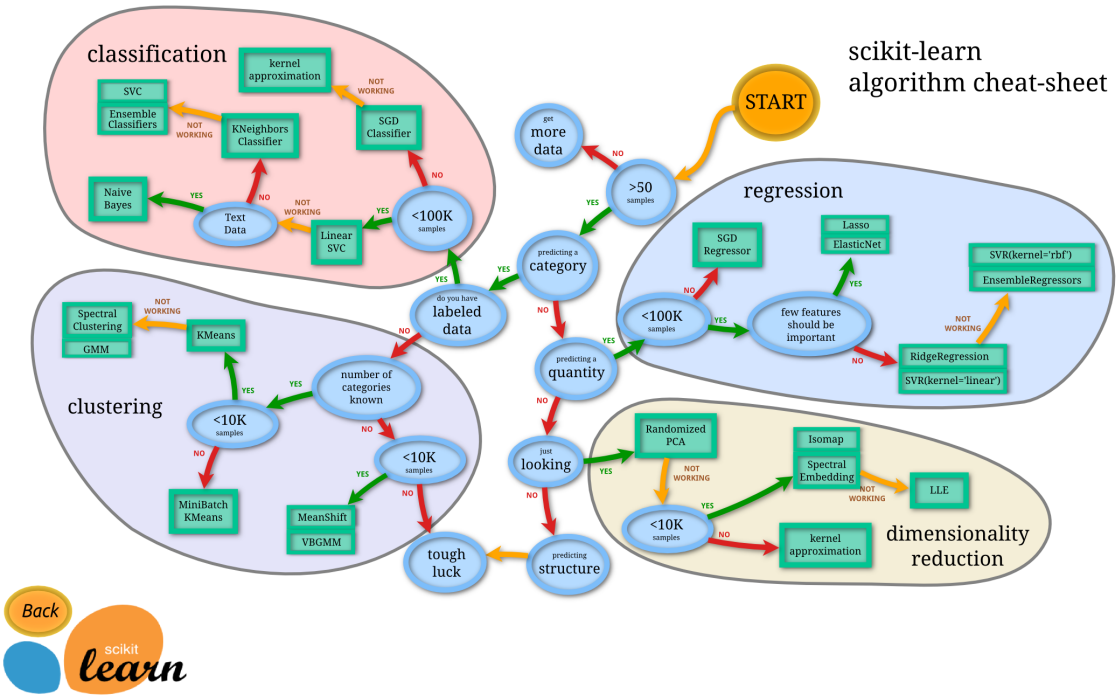






6 MODEL EVALUATION

Based on the EDA we analyze, we can follow the scikit-learn algorithm cheat-sheets to select our algorithm. Our prediction will be categorical(positive or negative). We choose the classification model to build our model. We also try popular alternative machine learning algorithms like XGBoost and BERT to have another view of our result.



6.1 SVC

Support vector classifier is a supervised learning model that could be used for classification. Its goal is to find the hyperplane or decision boundary that splits the dataset into groups. The optimal hyperplane has the maximum margin between the hyperplane and supporter vectors. Margin is the distance between the hyperplane and the nearest data points.

6.2 LinearSVC

Linear Support Vector Classifier (Linear SVC) is also a type of SVC that uses a linear kernel function to perform classification. It is similar to SVC with parameter kernel='linear', but the implementation behind it is different. It used a one-vs-the rest scheme. It performs well with a larger number of samples.

6.3 SGD

Stochastic Gradient Descent (SGD) is also a type of linear classifier. The difference is that the estimation of the SGD used gradient descent to optimize. It works best with dense or sparse matrices of floating-point values.

6.4 Naive Bayes

Naive Bayes is the most simple probabilistic classification based on applying Bayes theorem with a naive assumption that the features are independent of each other. Bernoulli Naive Bayes is specifically designed to be represented as a binary feature data and result.

6.5 Logistic Regression

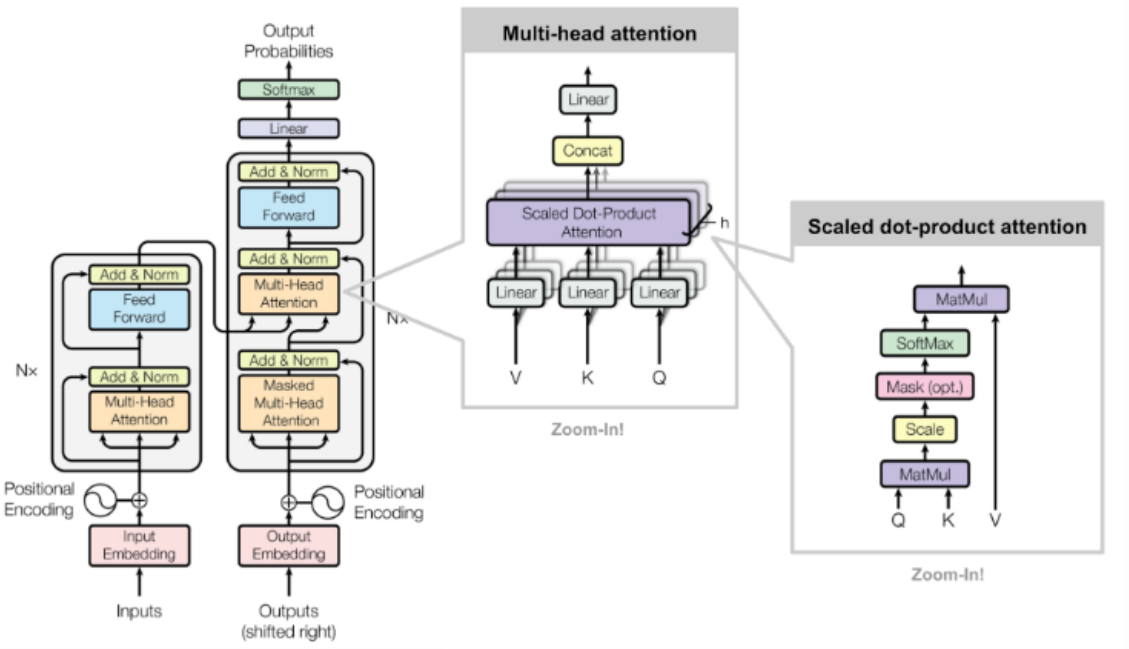
Logistic regression is a model that also finds the probability of a certain event such as pass/fail, win/lost/ alive/dead. Each object is detected in the image would be assigned a probability between 0 and 1. One difference that logistic regression has is that logistic does not assume that the data are independent of each other.

6.6 XGBClassifier

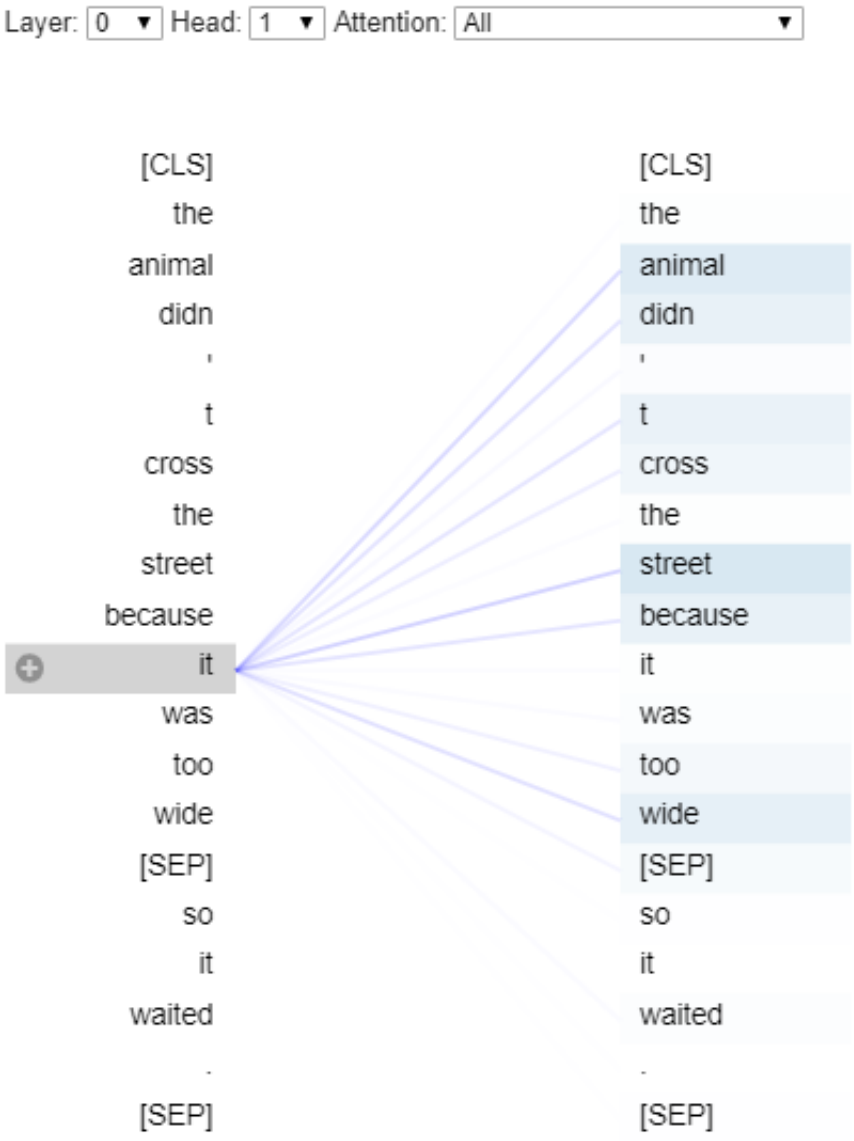
XGBoost is an algorithm that is an implementation of a gradient boosted decision tree designed for speed and performance optimization. The name XGBoost stands for eXtreme Gradient Boosting. XGBoosting is the most widely used algorithm for competition winners in Kaggle competitions.

6.7 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based deep learning model that was initially developed by Google for Natural Language Processing tasks. Transformers have the advantage of learning all contexts of words within texts. In comparison to directional models such as LSTM that only reads tokens from left to right or vice versa, transformers have an attention mechanism that allows tokens to have a certain weight for any other token in the form of Key, Query, and Value.



The model is trained through self-supervised learning which masks certain parts of the input sentence and has the model predict the blanks. The model is trained through self-supervised learning which masks certain parts of the input sentence and has the model predict the blanks.



BERT is trained on billions of texts across Wikipedia and other sources and the model itself has 110 million parameters. This allows the model to learn the structural information about language in depth. Because of this, we can take advantage of the pre-trained model and fine-tune it with an additional classifier layer to whatever task we choose.

In our case, we fine-tune the BERT model with a linear classifier for our binary labels (negative sentiment and positive sentiment) for sentiment classification. BERT is advantageous for language classification tasks since it does not require that much training data - since the pre-trained model already understands the majority of the semantic relationships between texts.

7 LITERATURE

This dataset has been studied by many other people on Kaggle before. Mostly, people perform sentiment analysis on this dataset. They used various methods or models to analyze what the text represents in people's tweets. It is very similar to our predictive task, so it is not a brand-new task to study. The difference between our project and others which work on a similar task is the way of data prepossessing and selection of models. In our data prepossessing, we made plenty of transformations to the text column to clean the data, such as expanding contractions, removing URLs, removing usernames, removing hashtags, removing punctuation, changing all letters to lowercase, etc. By cleaning this text, we can perform our analysis on more meaningful words so that we could explore whether the word represents negative or positive. We used the clean_text method to achieve this. However, other people did not clean the data as we did. Some of them just changed all the letters to lowercase then started to analyze the data. Besides, we used SVC, linear SVC, SGB, Naive Bayes, Logistic Regression, XGB Classifier, and BERT as our models. Other people did not use these models in their analysis. They only used two or three models to perform the sentiment analysis. The BERT model is the major novelty of our work because it is a concept that is not covered by the course and we have not seen many people using it in a similar task. Therefore, our conclusion will be a bit different from other people who are doing a similar task, since our BERT model has the highest accuracy among all the models we chose.

8 RESULTS

8.1 SVC

We tune the SVC model on the Sentiment140 dataset - size of 50000 The model was trained using verbose=True and cache_size=1000. It is mostly aid to speed up the training.
We chose accuracy as the evaluation of the models. The accuracy of the SVC model is 0.75

8.2 LinearSVC

We tune the LinearSVC model on the Sentiment140 dataset - size of 100000
The model was trained using liblinear with the default settings.
We do not see a significant difference in accuracy from training on 50,000 vs 100000. The accuracy of the SVC model is 0.74.

8.3 SGD

We tune the SGD model on the Sentiment140 dataset - size of 100000
The model was trained using the default settings.
We do not see a significant difference in accuracy from training on 50,000 vs 100000. The accuracy of the SGD model is 0.75.

8.4 Naive Bayes

We tune the Naive Bayes model on the Sentiment140 dataset - size of 100000
The model was trained using the default settings.
We do not see a significant difference in accuracy from training on 50,000 vs 100000. The accuracy of the Naive Bayes model is 0.74.

8.5 Logistic Regression

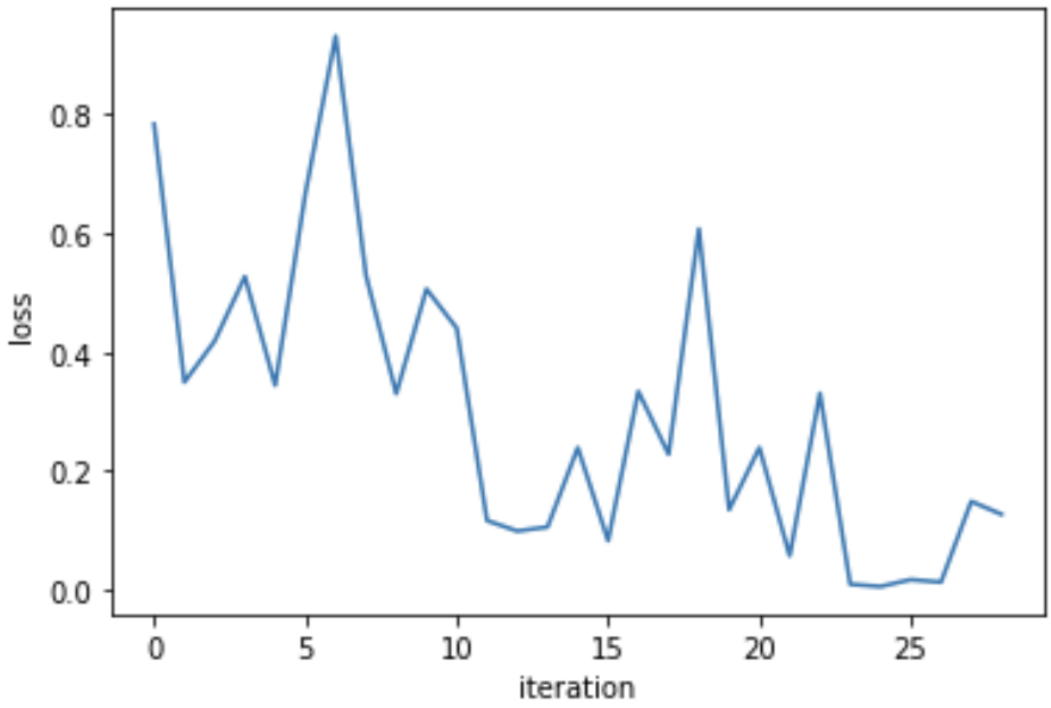
We tune the Logistic Regression model on the Sentiment140 dataset - size of 100000
The model was trained using the default settings.
We do not see a significant difference in accuracy from training on 50,000 vs 100000. The accuracy of the Logistic Regression model is 0.72.

8.6 XGBClassifier

We tune the XGBClassifier model on the Sentiment140 dataset - size of 100000
The model was trained using the default settings.
We do not see a significant difference in accuracy from training on 50,000 vs 100000. The accuracy of the XGBClassifier model is 0.73.

8.7 BERT

We fine-tune the BERT model on two sizes of the Sentiment140 dataset - size of 50,000 and size of 100,000.
The model was trained using AdamOptimizer with a learning rate of 5e-5 on a total of 3 epochs with a batch size of 8.



Where loss is cross-entropy loss and each iteration consists of 8000 examples.

Expectedly, we do not see a significant difference in accuracy from training on 50,000 vs 100,000. In the last epoch, the model scored an accuracy of 0.8247 for 50,000 examples and for 0.8257 for 100,000 examples - a mere 0.001 increase inaccuracy.

The test dataset consists of 10000 randomly sampled examples.

We clean the input data by simply removing usernames (ex: @abushedok). Our model with 50,000 examples scored an accuracy of 0.8381 on cleaned input and 0.8331 on uncleaned input data.

We observe that cleaning the data by removing usernames does not make a big difference. An explanation for this is that BERT is able to distinguish “@text” as a unique name and should not hold significant weight in the classification task.

9 SUMMARY

With the Twitter Sentiment Dataset, we aimed to create a machine learning model that can accurately classify tweets as positive sentiment or negative sentiment. This model can be used in many applications such as through AI assistants or anti-bully systems across social media.

Through data exploration, we observed the most common words in negative tweets and positive tweets. It was interesting to see the words "sad" and "miss" with negative tweets vs "thank" and "love" in positive tweets.

We performed a series of data exploration including removing unnecessary texts such as usernames as well as stemming words.

We tested multiple traditional ML models such as SVC, Naive Bayes, and Logistic Regression. We observe that the highest accuracy on our testing dataset is from LinearSVC and Logistic Regression with an accuracy of 0.77.

Lastly, we implemented a state-of-the-art BERT model which consists of a transformer neural network architecture. The main benefit of this model is that it is a large model consisting of hundreds of millions of parameters and pre-trained on billions of texts that allow the model to learn the structure of language. We fine-tune a classifier layer for our binary labels. We achieved an accuracy of 0.838 which is a significant increase compared to traditional ML models. This makes sense since the size of BERT is much larger and understanding language structure is crucial in classifying sentiments within a text.