TRANSFORMERS FOR SENTIMENT ANALYSIS (BERT)

Team 8

Sravani Somepalli

Anupama Pogaku

Phillip Zhu

Yiheng Zhang

Wenke Yu

Introduction to Sentiment & Big Data Analysis

We use sentiment analysis to get the public's evaluation and attitude of tweets and use the data analysis to make business decisions accordingly. As Social media is increasing its traffic day by day, it is always an advantage to businesses to monitor the customer's opinions constantly. Taking movie reviews as an example, since anyone can review movies in any form on social media, and these reviews can be seen and disseminated by any user, prioritizing the availability of these reviews helps many significant business decisions, including but not limited to the production, promotion, and marketing of movies, cinema scheduling, design and sales of related merchandise, etc. Therefore, we choose this topic and devote ourselves to making valuable business suggestions by the means of machine learning and data analysis.

Justification

Sentiment analysis is especially important because it can be a company's only source of truth about what people are trying to communicate with it. Taking the Movie Industry as an example. For a movie, on the one hand, the audience's review will have a huge impact on it in the commercial aspect. A film production company can understand the current market trends based on reviews. In order to attract more investors, film production companies can determine the subject and type of future films through evaluation. And investors can also use reviews to understand what types of movies the audience likes to watch to decide what movies to invest in next. On the other hand, some discussions in the reviews can help insiders to locate and evaluate the artistic value of the movie. In the fierce industry competition, it is an arduous task to establish and maintain the reputation of a movie. With the widespread use of sentiment analysis, film production companies can more easily monitor public opinion trends. In summary, the industry needs to analyze movie reviews through sentiment analysis. The more the industry uses sentiment analysis, the more positive development can bring to the industry.

End Goals:

There are a couple of questions that we are attempting to answer from the data:
How can we analyze customer sentiment based on their tweet?
How accurately can we analyze customer sentiment based on their tweet?
What are some trends we notice regarding the positive vs negative tweets that we predict?
From our findings, what are some of the real-world applications that would be useful for a business?
Through the valuable insights we achieve, enterprises can carry out emotional packaging, emotional promotion, emotional advertising, emotional word of mouth, emotional design, and other strategies to achieve product business goals.

Introduction to Transformers and BERT:

Google introduced Transformers in 2017. At the time of their introduction, recurrent neural networks (RNN) and convolutional neural networks (CNN) are the primarily used language models for NLP tasks. Though they are quite competent transformers were considered revolutionary because unlike CNN and RNN, transformers don't need the data to processed in sequence and they can also train the data in larger amounts than before. Later in 2018, Google released open-sourced BERT, which was an already trained model on massive amounts of language data prior to its release. Even in research stages, BERT achieved groundbreaking results in 11 natural language understanding tasks like sentiment analysis, semantic role labeling, sentence classification etc.,

Computation:

We ran this entire pipeline end-to-end using PyTorch since we plan to use GPUs for this computation. We are using Google Colab to run our dataset. We are moving the runtime to GPU since we will be processing text data and it is difficult for a CPU to handle it.

Dataset Description:

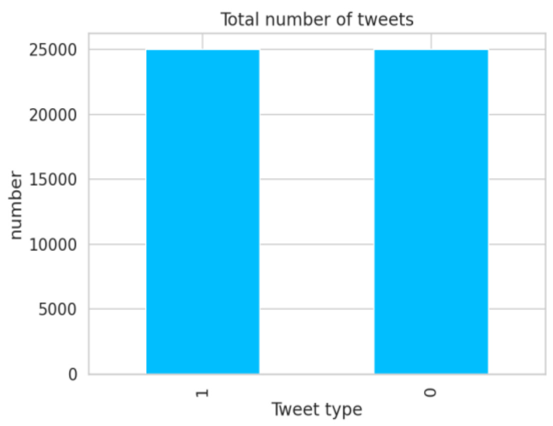
We are using Kaggle as our resource to collect the tweets data to proceed with sentiment analysis. The original Shape of the data - (1600000, 6) - 1.6 million records spread across 6 columns. However due to compute constraints, we decided to go with 50,000 records.

- ❖ The dataset have columns named as following - Sentiment, ID, Date , Query, User, Tweet
- ❖ We uploaded the data into Google colab using the upload function and read the data into pandas dataframe(df). We read the data into another dataframe(df2) to use for our final tests.

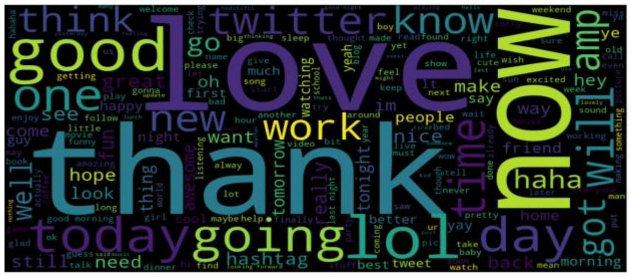
Exploratory Data Analysis:

We performed Exploratory analysis(EDA) to analyze the parameters at the beginning of the project.

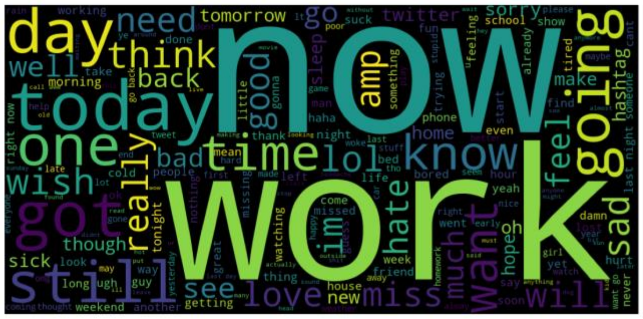
- ❖ The shape of the data we used to build the model is (50000, 6). 50,000 records with 6 columns.There are no null values in any of the columns and we verified it.
- ❖ We tried to understand the data better by finding the number of unique users and the dates that received most tweets.
- ❖ Number of users involved in the data -> 41237
- ❖ Number of days tweets were done ->28429
- ❖ Most Tweets done on -> Fri Apr 17 21:57:42 PDT 2009
- ❖ User who has done most tweets ->sebby_peek
- ❖ Tweet distribution before splitting the data into Test Train and Validation sets is as follows.



- We wanted to showcase the words used the most in the positive tweets. All the positive tweets(Sentiment =1) were copied into a new dataframe to build the below wordcloud.



- We did the same thing for the negative tweets and the wordcloud showed up as below.



BERT for Sentiment Analysis:

To use a pre-trained model like BERT, we converted the input data into proper format so that each sentence can be sent to BERT.

We are using PyTorch library to train and evaluate the models. We imported the Transformer model (BERT) from Hugging Face since it is a specialized library for Natural Language Processing. We will be using the pre-trained model and fine-tune the model on our data.

- We decided to go with the pretrained BERT variant 'bert-base-cased' since it is highly effective, case sensitive and able to differentiate between the sentences. We are going to fine tune it on our data and generate the model with best accuracy.

- We removed the existing last layer of BERT and replaced it with a linear layer which outputs two logits since our classification problem has two classes.
- We installed the transformers and imported the required packages to load the model.

Tokenization:

Since ours is a classification task, a single vector that represents the whole sentence, needs to be fed to the classifier

[CLS] Token:

- In BERT, the first token represents the hidden state of entire sentence. We additionally/manually added [CLS] token to the input sentence.

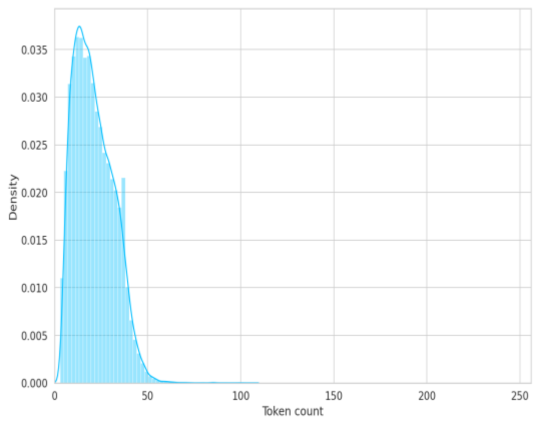
[SEP] Token:

- When we are training the classifier, each input sample will contain only one sentence. Hence we add a [SEP] token at the end of each input text.

[PAD] Token:

- Once we set a maximum length of the tweet, the empty spaces will be padded using the [PAD] token.

- We used the tokenizer to convert the text into tokens that can be fed to the model. We started with a sample tweet and converted the text into tokens using the tokenizer. The maximum length was taken as 70 for the sample tweet and the tokenizer padded the empty spaces. The same logic was applied on the rest of the tweets and the data was moved into token_lens which consists of the length of tokens of each tweet.
- The function will pad the empty spaces.
- According to the seaborn plot, most of the tweets are below 100 tokens. So, we decided to go with the max length 100.



Train, Validation and Test sets:

- We used scikit-learn train_test_split to split the dataset into train, validation and test sets.
- Training data is populated with 45000 records and the validation and test sets are populated with 2500 records each.

DataLoader:

- Used PyTorch DataLoader that feeds the data in batches based on the batch size.
- All three dataloaders were created for training, validation and test sets.
- The number of workers is set to '0' since we ran the code on the Colab GPU and did not use the CPUs.

SentimentClassifier:

- The sentiment classifier class helps the model to input the ids and drop the irrelevant ones based on the number of classes(in our case we have 2 classes)

Softmax:

- The softmax function of PyTorch helps us find out the probabilities based on the batch size and number of classes that rescales them between 0,1. In this case the batch size was taken as 64 and softmax was able to provide the matrix with (64 X 2) dimensions.

EPOCHS:

- Epoch stands for every one successful run through the data. The DataLoader will run many iterations based on the batch size and it is considered as an Epoch once the DataLoader is done passing all the data.
- Mathematically speaking, our training data have 45000 records and the batch size is 64. So every iteration will have 64 records and the results in 45000/64 iterations for every epoch.

AdamW:

- AdamW is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data
- We used this stochastic optimization method to reduce the loss function.

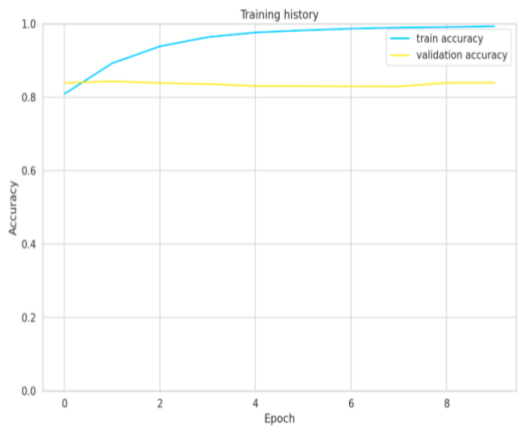
Training and Validation set results:

- After the 10 epochs, the training and validation sets showed the below results

Epoch	Train accuracy	Val accuracy	Train loss	Val loss
1	0.8086888888888889	0.8388	0.4167840741574764	0.3632372707128525

2	0.8921777777777778	0.8432000000000001	0.2635022141674364	0.38248375803232193
3	0.9384	0.8384	0.16290248947916552	0.46805646643042564
4	0.9635333333333334	0.8356	0.10400453186924676	0.5857437692582608
5	0.9762222222222223	0.8304	0.07439704651782945	0.6992280505597591
6	0.9817777777777779	0.8300000000000001	0.0578375550709428	0.7369212344288826
7	0.9865333333333334	0.8296	0.045090600275216275	0.8571939878165722
8	0.9896	0.8292	0.03514857164459913	0.9323008388280869
9	0.9909555555555556	0.8388	0.029641588782252173	0.9833269201219081
10	0.993	0.8392000000000001	0.023073247748966838	1.0168792724609375

- ❖ The model with best accuracy will be considered to apply on test data. It is loaded into the device.
- ❖ The visual representation of training data and validation data accuracies are shown below.

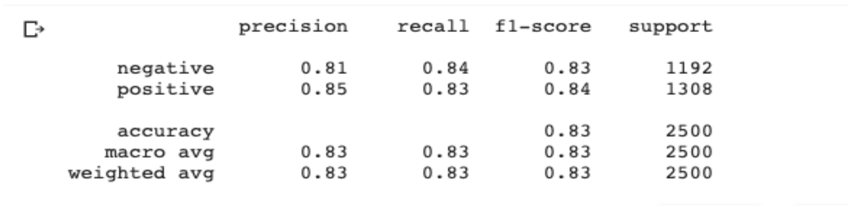


Applying model on test set:

- The model is applied on a test set to generate the test accuracy. Here the test accuracy yielded to 0.8312 which is almost equal to the best accuracy of validation set(0.8432000000000001)
- The test set is set to get the predictions. The predicted values are moved to y_tweets(which has the text of the tweets from dataset) , y_pred(The predicted sentiment), y_pred_probs(the prediction probabilities), y_test(the actual values of sentiment)

Classification report:

- The classification report presents the precision recall and f1 score of the predicted values.



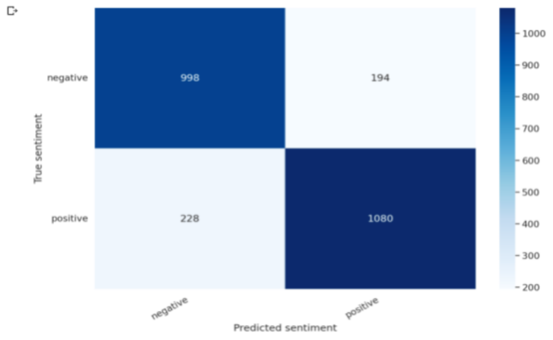
```
              precision    recall  f1-score   support

    negative       0.81      0.84      0.83      1192
    positive       0.85      0.83      0.84      1308

    accuracy                           0.83      2500
   macro avg       0.83      0.83      0.83      2500
weighted avg       0.83      0.83      0.83      2500
```

Visual Representation of Confusion matrix:

- Confusion matrix displayed the actual values



True Positive: 1080

True Negative: 998

False Positive: 194

False Negative: 228

Predicted values Dataframe:

To experiment with our predicted values, we created a new dataframe with the predicted sentiment values and the tweets.

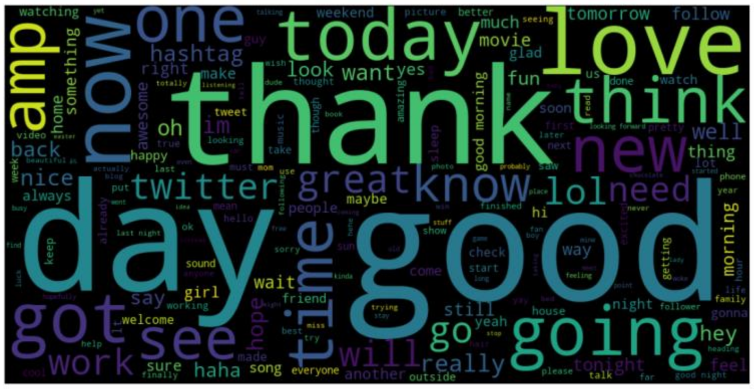
Another dataframe with Tweet and Original values(same as test data).

The dataframe pred_df has all the records of the test set with the tweets and the predicted sentiment.

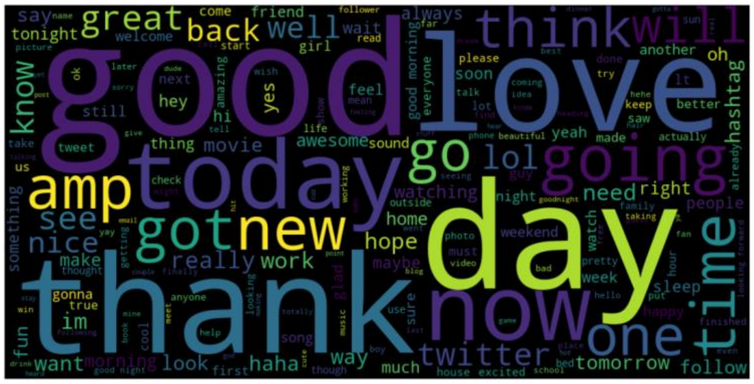
We went ahead and separated the dataframes into two new dataframes each with positive and negative values.

The new dataframes were used to generate the wordclouds of positive and negative tweets and they look quite similar.

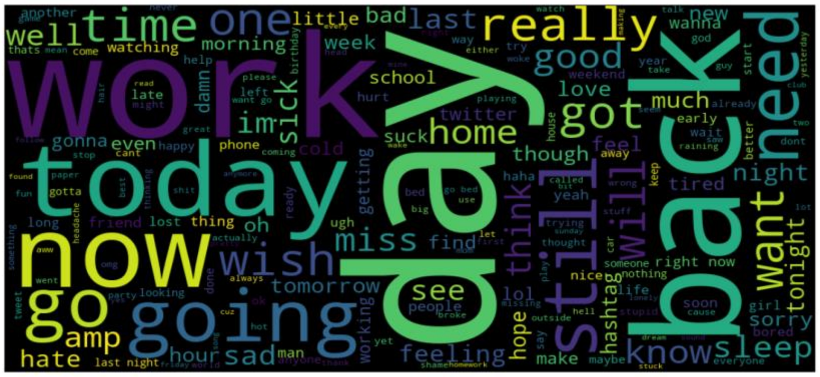
Positive wordcloud original values:



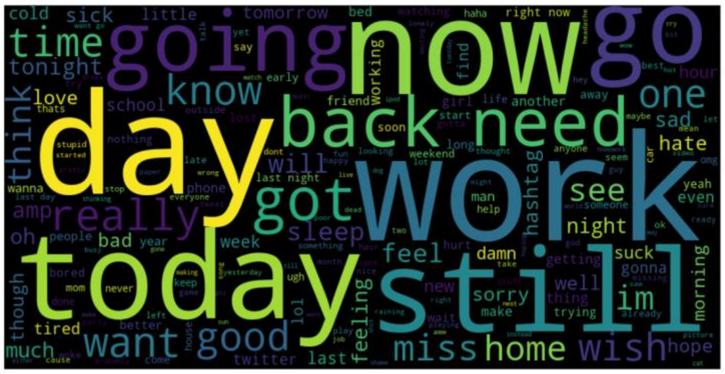
Positive wordcloud predicted values:



Negative wordcloud with original values:

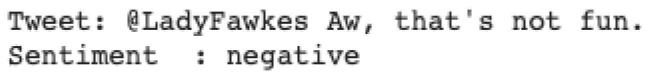


Negative wordcloud with predicted values:

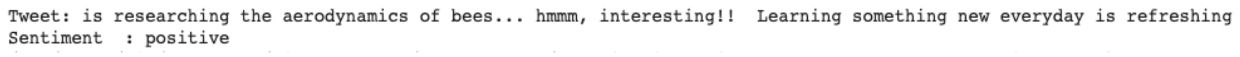


Predicting the sentiment of the tweet that the model never saw:

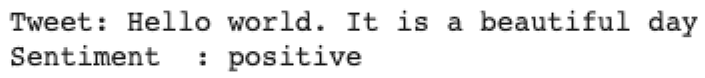
Using the tokenizer and the fine tuned model, we sent new text that was not part of training, validation or test sets and the model can predict the sentiment accurately.



```
Tweet: @LadyFawkes Aw, that's not fun.
Sentiment  : negative
```



```
Tweet: is researching the aerodynamics of bees... hmmm, interesting!!  Learning something new everyday is refreshing
Sentiment  : positive
```



```
Tweet: Hello world. It is a beautiful day
Sentiment  : positive
```

Conclusion:

Using a pre-trained model for our analysis indeed reduced a lot of work and in addition to it we were able to utilize the best model by running multiple epochs. The fine-tuned model can be downloaded locally and we can utilize it on the new data in future.

This model can be used not just to predict the tweet sentiment but also the product review or movie review sentiment.