Sentiment Analysis

Made By :- Siddhant Mane

Objective :-

- Develop a sentiment analysis model using Tensorflow that assigns a sentiment looking at the review/chat of the product.
- Use pre-built embeddings for your data dictionary
- Train the model using Transformer/Attention based architecture
- Reach an overall “precision” score of 85%
- Use TF serving to deploy the model as an API (on local)
- Build a TF serving client to interact with the API. This client should also be able to continuously accept data entered by the user and provide the sentiment for the review/chat entered by the end-user.

The Dataset

The dataset being used is the sentiment140 dataset. It contains 1,600,000 tweets extracted using the Twitter API. The tweets have been annotated (0 = Negative, 4 = Positive) and they can be used to detect sentiment.

It contains the following 6 fields:

sentiment: the polarity of the tweet (0 = negative, 4 = positive)

ids: The id of the tweet (2087)

date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)

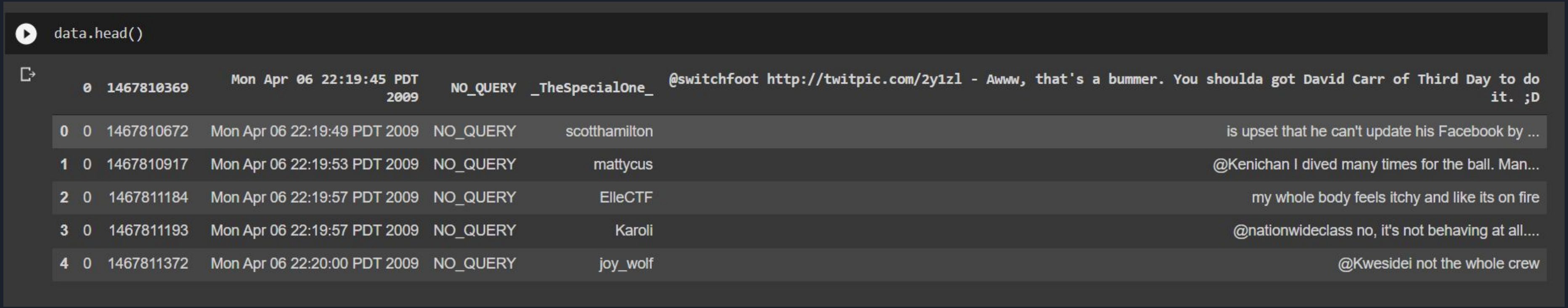
flag: The query (lyx). If there is no query, then this value is NO_QUERY.

user: the user that tweeted (robotickilldozr)

text: the text of the tweet (Lyx is cool)

We require only the sentiment and text fields, so we discard the rest.

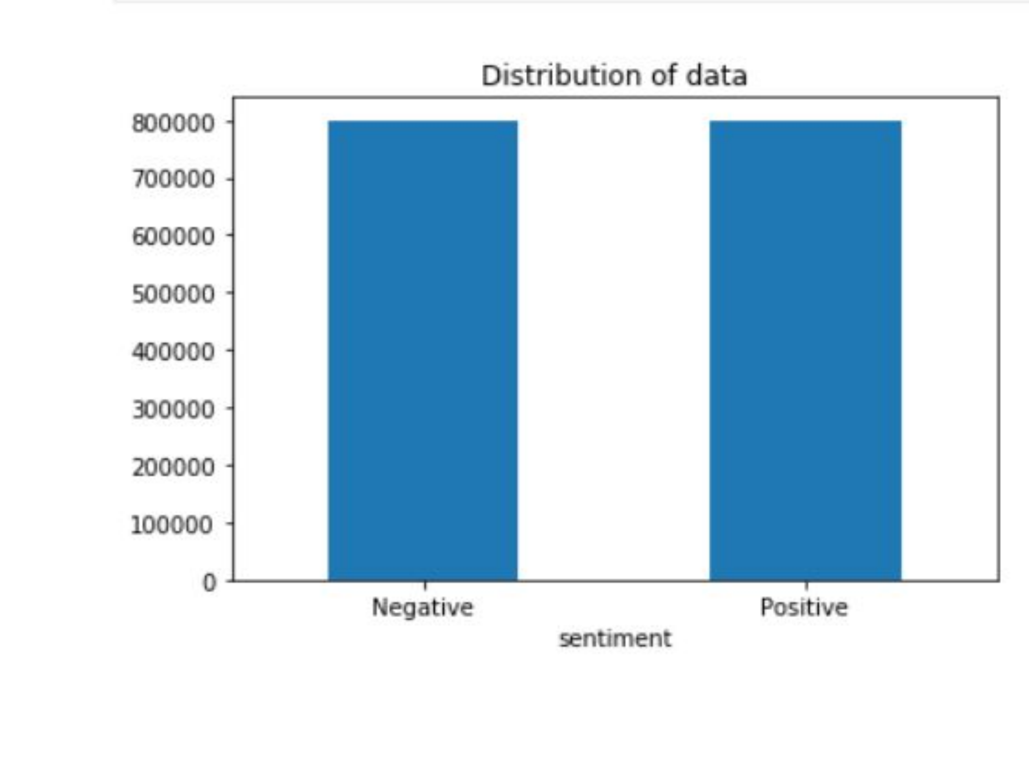
Furthermore, we're changing the sentiment field so that it has new values to reflect the sentiment. (0 = Negative, 1 = Positive)



```
data.head()
```

	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

First 6 Rows of Dataset



There is no Imbalance in the dataset as value counts of both Negative and Positive Sentiments are same

Preprocessing of Text :

Lower Casing: Each text is converted to lowercase.

Replacing URLs: Links starting with "http" or "https" or "www" are replaced by "URL".

Replacing Emojis: Replace emojis by using a pre-defined dictionary containing emojis along with their meaning. (eg: ":)" to "EMOJIsmile")

Replacing Usernames: Replace @Usernames with word "USER". (eg: "@Kaggle" to "USER")

Removing Non-Alphabets: Replacing characters except Digits and Alphabets with a space.

Removing Consecutive letters: 3 or more consecutive letters are replaced by 2 letters. (eg: "Heyyyy" to "Heyy")

Removing Short Words: Words with length less than 2 are removed.

Removing Stopwords: Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. (eg: "the", "he", "have")

Lemmatizing: Lemmatization is the process of converting a word to its base form. (e.g: “Great” to “Good”)

Encoding of Data :

Tokenizer :

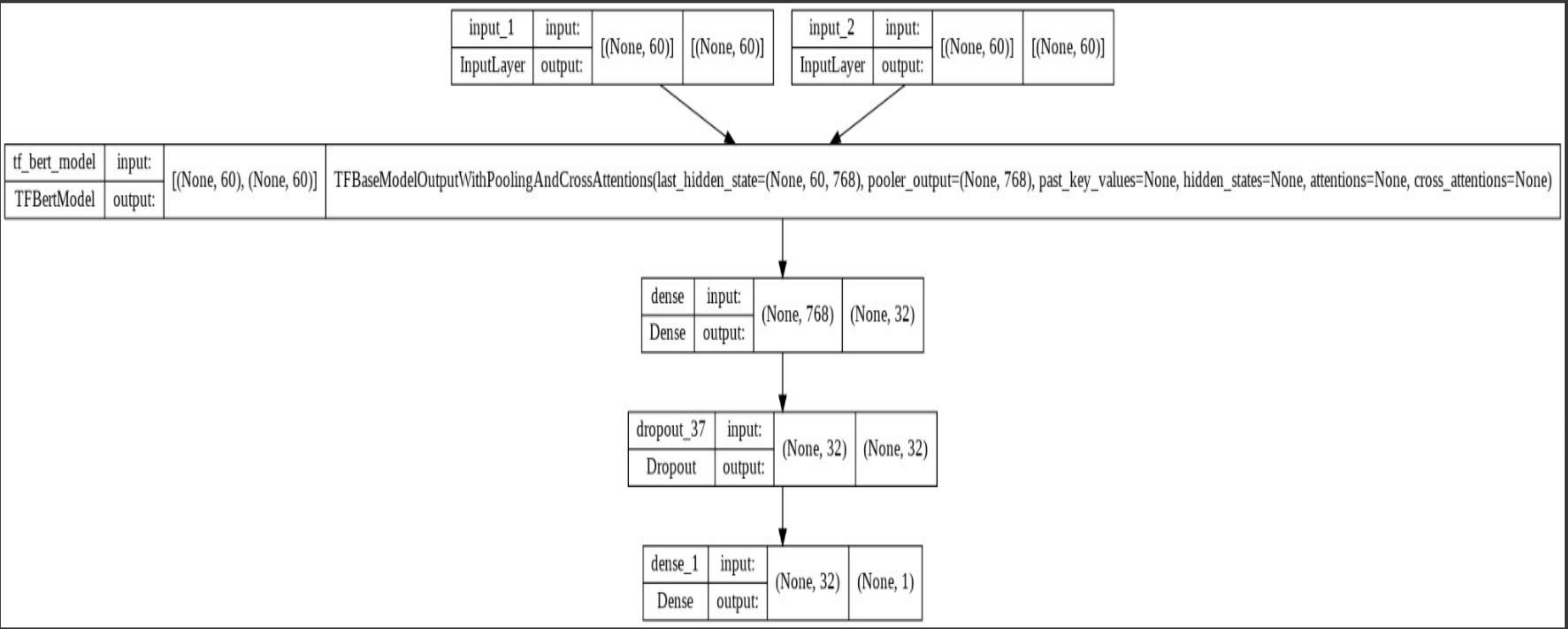
Natural language processing is one of the fields in programming where the natural language is processed by the software. This has many applications like sentiment analysis, language translation, fake news detection, grammatical error detection etc.

The input in natural language processing is text. The data collection for this text happens from a lot of sources. This requires a lot of cleaning and processing before the data can be used for analysis.

4 methods:

1. Tokenizer
2. Encode_plus
3. Encode
4. Tokenize and then get_token_ids

Model



input_1 | input: | [(None, 60)] | [(None, 60)]
InputLayer | output:

input_2 | input: | [(None, 60)] | [(None, 60)]
InputLayer | output:

tf_bert_model | input: | [(None, 60), (None, 60)] | TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 60, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None)
TFBertModel | output:

dense | input: | (None, 768) | (None, 32)
Dense | output:

dropout_37 | input: | (None, 32) | (None, 32)
Dropout | output:

dense_1 | input: | (None, 32) | (None, 1)
Dense | output:

Results after fitting the model :

After First Epoch :

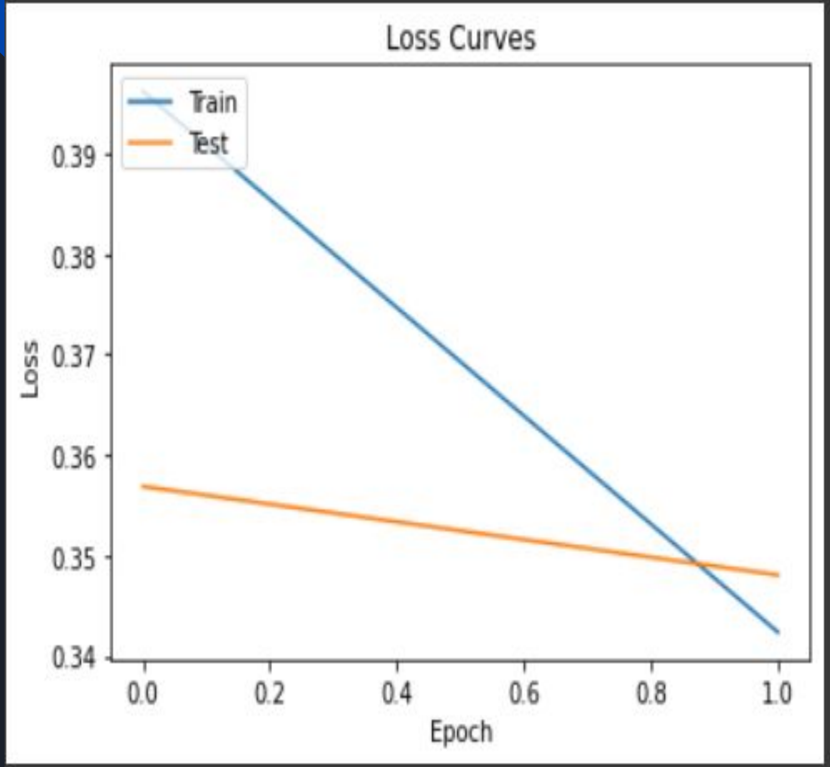
loss: 0.3962 - accuracy: 0.8213 - val_loss: 0.3569 - val_accuracy: 0.8430

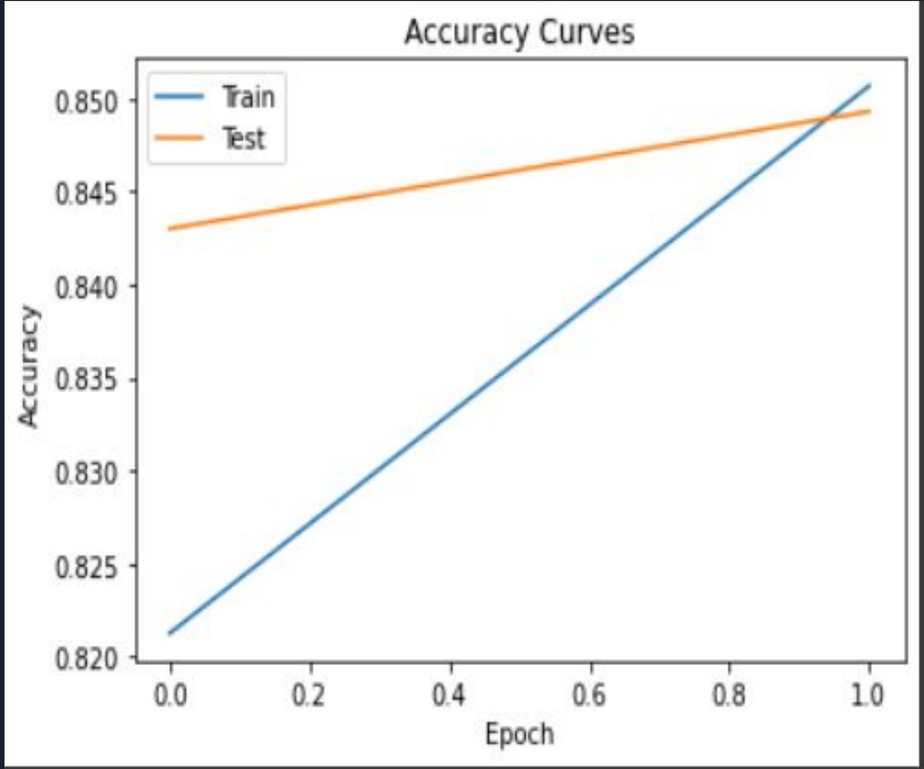
After Second Epoch :

loss: 0.3424 - accuracy: 0.8507 - val_loss: 0.3481 - val_accuracy: 0.8493

Able to get 85% validation accuracy after 2 epochs

Curves :


Loss Curves
Train
Test
Loss
0.39
0.38
0.37
0.36
0.35
0.34
0.0
0.2
0.4
0.6
0.8
1.0
Epoch


Accuracy Curves
Train
Test
Accuracy
0.850
0.845
0.840
0.835
0.830
0.825
0.820
0.0
0.2
0.4
0.6
0.8
1.0
Epoch