CAPSTONE REPORT

Tweet Sentiment Extraction

1. Definition

Project Overview

Microblogging websites like Twitter are platforms for the expression of opinions in real-time to millions of users. These tweets can potentially impact a person, brand, company, or country due to the freedom of expression, and the wide reach one can potentially obtain on the platform. A lot of effort and research [1] [2] [3] [4] has gone into understanding the sentiment of millions of tweets being made by the varied users of the platform. Running a sentiment analysis machine learning model allows brands/companies to monitor the sentiment of their campaign in real-time, thus allowing them to respond effectively to any negative sentiment on the rise.

I've built various sentiment analysis models in both professional and academic capability, most recently as part of Udacity's Machine Learning Nanodegree. This proposal is my attempt to go a step deeper into this domain.

Problem Statement

We have seen phenomenal growth in the accuracy of predictions by a sentiment analysis model in the past couple of years. However, a person must still go through the model predicted sentiments to capture patterns of words that reflect the sentiment. A machine learning model that captures the words in a tweet that support a positive, negative, or neutral sentiment can work in tandem with a regular sentiment analysis model.

Kaggle has created a competition for this task by offering the 'Sentiment Analysis: Emotion in Text' dataset from Figure Eight's Data for Everyone platform [5]. Thus, the objective is to look at the labeled sentiment for a given tweet and figure out what word or phrase best supports the sentiment.

As part of preprocessing the data, I intend to tokenize the words using Bert Word Piece tokenizer from Huggingface [7]. I might also look at removing punctuations, stopwords as they are unlikely to have any effect on the predicted sentiment. For this analysis, I intend to do EDA on the 'selected_text' labels to see if they have any punctuations and stopwords among them.

For model training purposes, I intend to leverage transfer learning by stacking pre-trained weights from models like BERT, RoBERTa onto my custom layers. Keeping in mind the objective, it deals with a Question-Answering task, so I'll use the QuestionAnswering versions of the Huggingface BERT model.

Metrics

The evaluation metric as defined in the Kaggle competition [5] is the word-level Jaccard score. The Jaccard score between two strings can be calculated using the following Python code snippet:

```python
def jaccard(str1, str2):
    a = set(str1.lower().split())
    b = set(str2.lower().split())
    c = a.intersection(b)
    return float(len(c)) / (len(a) + len(b) - len(c))
```

The Jaccard score calculates intersection over union and is a good score to measure the similarity between finite sample sets.

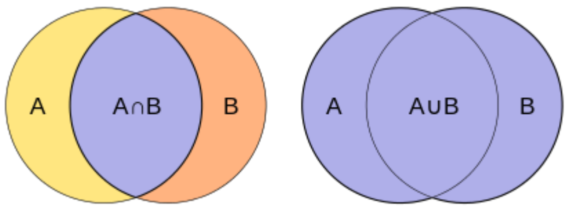


Fig. 1 Intersection and union two sets A and B

When we predict a ‘selected_text’, we want to as approximate to the true label as possible. The goal is to capture all of the true labels i.e. maximize the intersection, and at the same time reduce false-positive i.e. minimize the union Thus, Jaccard similarity is an appropriate metric for this problem.

The overall evaluation metric for all tweets in the test set is:

$$score = \frac{1}{n} \sum_{i=1}^{n} jaccard\left(gt_i, dt_i \right)$$

$$\begin{aligned}
&where,\\
&n = number\ of\ documents\\
&jaccard = the\ function\ provided\ above\\
&gt_i = the\ ith\ ground\ truth\\
&dt_i = the\ ith\ prediction
\end{aligned}$$

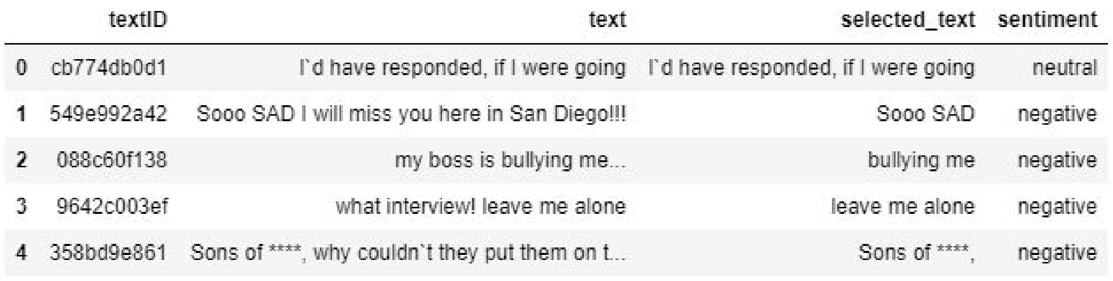
2. Analysis

Data Exploration

As mentioned in the Problem Statement above, I'll be using the 'Sentiment Analysis: Emotion in Text' dataset [5]. This dataset is available for use under Creative Commons Attribution 4.0. International license. The dataset is split into training and test set. Being part of active competition, the test set does not have true labels.

The data is saved in a comma-separated file. Each row contains the 'text' of the tweet and a 'sentiment' label. In the training set, we have the 'selected_text'; that contains a word or phrase drawn from the tweet that encapsulates the provided sentiment. The 'selected_text' field is what the trained machine learning model will be predicting. Here is a sample of the training dataset:

```
train.head()
```



	textID	text	selected_text	sentiment
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
2	088c60f138	my boss is bullying me...	bullying me	negative
3	9642c003ef	what interview! leave me alone	leave me alone	negative
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative

We won't require the column 'textID' for our model training here. There are 3 unique values of sentiment: positive, negative, and neutral.

```
train.isna().sum()
```

```
textID           0
text             1
selected_text    1
sentiment        0
dtype: int64
```

The dataset has only one 'textID' with NaN value for the columns 'text', and 'selected_text'. I will drop this entry in the data preprocessing stage.

The training dataset will have a total of 27480 data points after dropping the NaN entry. The test dataset has a total of 3534 data points with no NaN values in it.

Exploratory Visualization

The dataset has tweets categorized into 3 sentiments - positive, negative, and neutral. Before choosing an algorithm, I wanted to explore and answer the following two questions:

1. Is this a balanced dataset?
2. Is the distribution of the test dataset the same as the train dataset?

This is the result of my analysis:

Fig. 2 *Count of sentiments in training data*

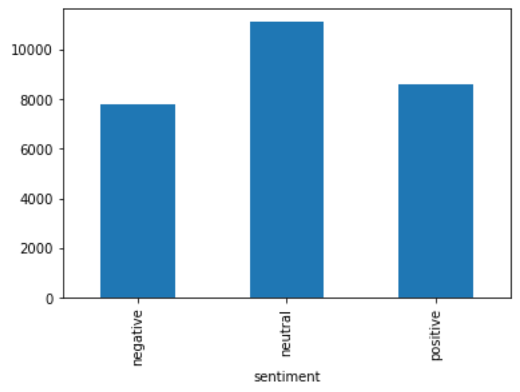
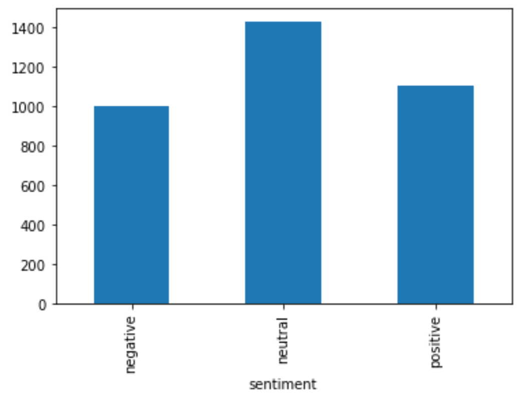


Fig 3. *Count of sentiments in testing data*



It can be seen from the above plots that the dataset is mostly a balanced dataset. However, I'm still going to make use of Stratified cross-validation to ensure that the model is a representative of each sentiment. The distribution of train and test dataset is also similar and thus, this allows flexibility in algorithm choice.

It can be seen from the training data sample, that there are instances where the 'selected_text' is the same as 'text'. Let's do an analysis of such tweets:

Fig 4 selected_text == text in Orange

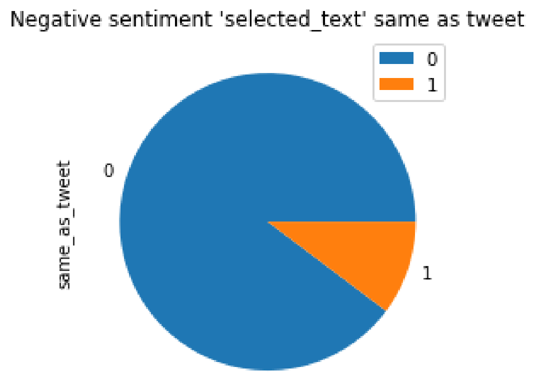


Fig 5 selected_text == text in Orange

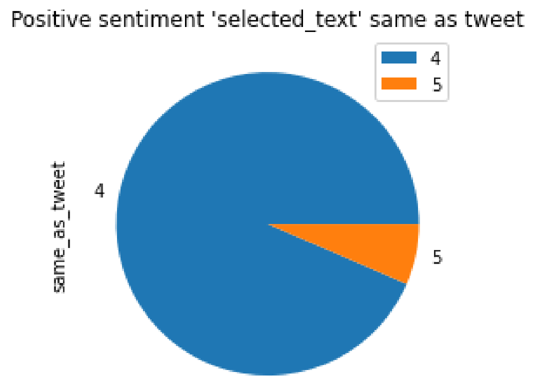
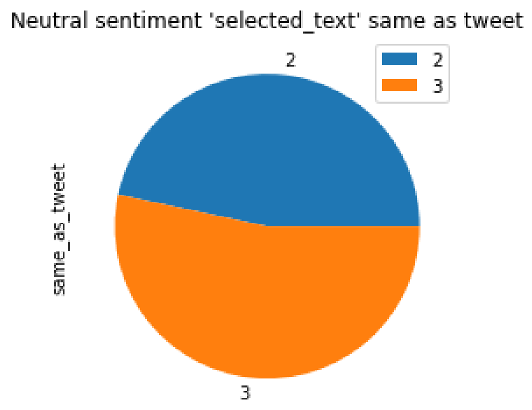


Fig 6 selected_text == text in Orange



We can see from these pie charts, that for Neutral sentiment, more than 50% of 'selected_text' is same as 'text' (tweet)

Algorithms and Techniques

For this Question Answering task, the algorithm is going to be RoBERTa which is a bidirectional transformer that builds upon BERT. For model training purposes, I intend to leverage transfer learning by stacking pre-trained weights from RoBERTa onto my custom layers.

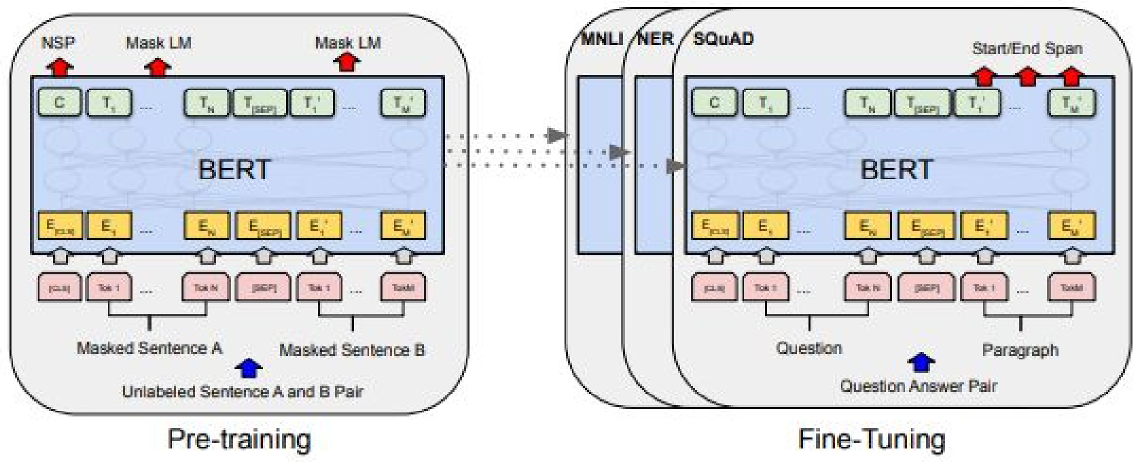


Fig. 7 Overall pre-training and fine-tuning procedures for BERT/RoBERTa. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning (which we'll be performing), all parameters are fine-tuned.

BERT architecture remains unified across different tasks. The pre-trained architecture on the left in Figure 7, with minimal modifications, can be adapted to different downstream tasks like Question Answering. This is the biggest advantage of BERT/RoBERTa.

As part of preprocessing the data, I intend to tokenize the words using Byte Level BPE tokenizer from Huggingface [7]. BPE is a greedy algorithm, that starts with unique characters and their frequency in the vocabulary. It tries to find a way to represent the entire text dataset with the least amount of tokens by merging the characters. Merging in a BPE algorithm is done by identifying the most frequently represented byte pairs. BPE tokenizer results in subword tokenization of both seen and unseen words.

The following parameters can be tuned to optimize the model:

- Training parameters:
  - Number of folds (in Stratified CV)
  - Number of epochs
  - Batch size
  - Optimizer type
  - Learning rate
- Custom layers on top of RoBERTa
  - Number of layers
  - Layer types (Conv1D, Flatten, Dropout)
  - Layer parameters
- Preprocessing parameters
  - Max token length
  - Case sensitive/insensitive

I use a GPU compute instance for model training and I create model checkpoints for the best validation loss. The saved model weights are loaded in a separate inference notebook, which can be submitted on Kaggle to get the score on Leaderboard.

Benchmark

For a baseline model, I went over some of the submissions on Kaggle and adapted a simple solution [6] that achieved a score of 0.651 on the Kaggle leaderboard. This solution creates a dictionary of words for each sentiment group where the values are the proportion of tweets that contain those words. Afterward, this algorithm is applied to a tweet to extract words or phrases describing the tweet. My solution will try to beat this score, and I describe the approach I intend to take below.

3. Methodology

Data Preprocessing and Transformation

The preprocessing done in the “tweet-extraction_train” notebook consists of the following steps:

1. Being part of a Kaggle competition, the dataset is relatively clean with only one data point with NaN values which can be dropped without any repercussions.
2. Both punctuations and stopwords occur frequently in the ‘selected_text’, and thus I decided to keep them in the dataset.
3. Being an NLP task, I need to create a numerical mapping of the textual data in the form of numerical vectors. These numerical vectors must be able to preserve the semantic and syntactic relationships in the text. The RoBERTa model was trained with byte-level BPE tokenization.
4. As part of the tokenization, I’ve converted all characters to lowercase and have added a prefix space before each token.
5. I needed to decide on a *max_length* of tweet text to create a homogeneous shape of vectors for each tweet. Fortunately, this is easy for this problem due to the maximum tweet length imposed by Twitter, currently 280 characters. This means that the maximum number of possible worlds can be 140 (when every word is a single letter). I pad all the vectors of tweets that are not of this length.

There are also some preprocessing and transformation steps which are performed before we feed data to the RoBERTa model:

1. Input IDs - They are token indices, numerical representations of tokens building the sequences that will be used as input by the model.
2. Attention Mask - This indicates to the model which tokens should be attended to, and which should not.
3. Token Type IDs - RoBERTa model does not require these, but we have to pass an array of zeros in the Huggingface implementation of RoBERTa.

4. Start Tokens (labels) - This indicates the start of the ‘selected_text’ in the Input IDs

5. End Tokens (labels) - This indicates the end of the ‘selected_text’ in the Input IDs

In the training dataset, there were 156 tweets containing Unicode characters. The Huggingface ByteLevelBPE tokenizer has trouble handling the Unicode characters and it couldn’t decode them correctly. In my implementation, as a result, the Input IDs, Start/End tokens are incorrectly calculated. Being a small sample, this doesn’t adversely affect the RoBERTa model.

Implementation

Data Modeling

The authors of RoBERTa[8] (released in July 2019) state that BERT was significantly undertrained, and can match or exceed the performance of every model published after it. BERT[9] was trained for four days on 4 to 16 cloud TPUs. Since, I do not have similar resources at hand, the quality of pre-training is important and I chose RoBERTa instead of BERT.

The Huggingface implementation of RoBERTa, however, does not have a Question Answering Head unlike their BERT implementation. I use a custom Question Answering head[10] on top of RoBERTa here.

1. The RoBERTa model takes Input IDs, Attention Masks, Token Type IDs as input.

2. The bare RoBERTa model transformer outputs raw hidden-states of the shape (batch_size, TOKEN_MAX_LEN, 768).

3. The custom Question Answering head takes this output, applies a Conv1D layer to transform the embedding to the shape (batch_size, TOKEN_MAX_LEN, 1).

4. This is then flattened and a Softmax function is applied to get the output of shape (batch_size, TOKEN_MAX_LEN).

5. These are one-hot encodings of the start token indices of ‘selected_text’. Similarly, the Conv1D, Flatten, and Softmax is applied to the raw output to get the end token indices of ‘selected_text’.

The model architecture will explain this further:

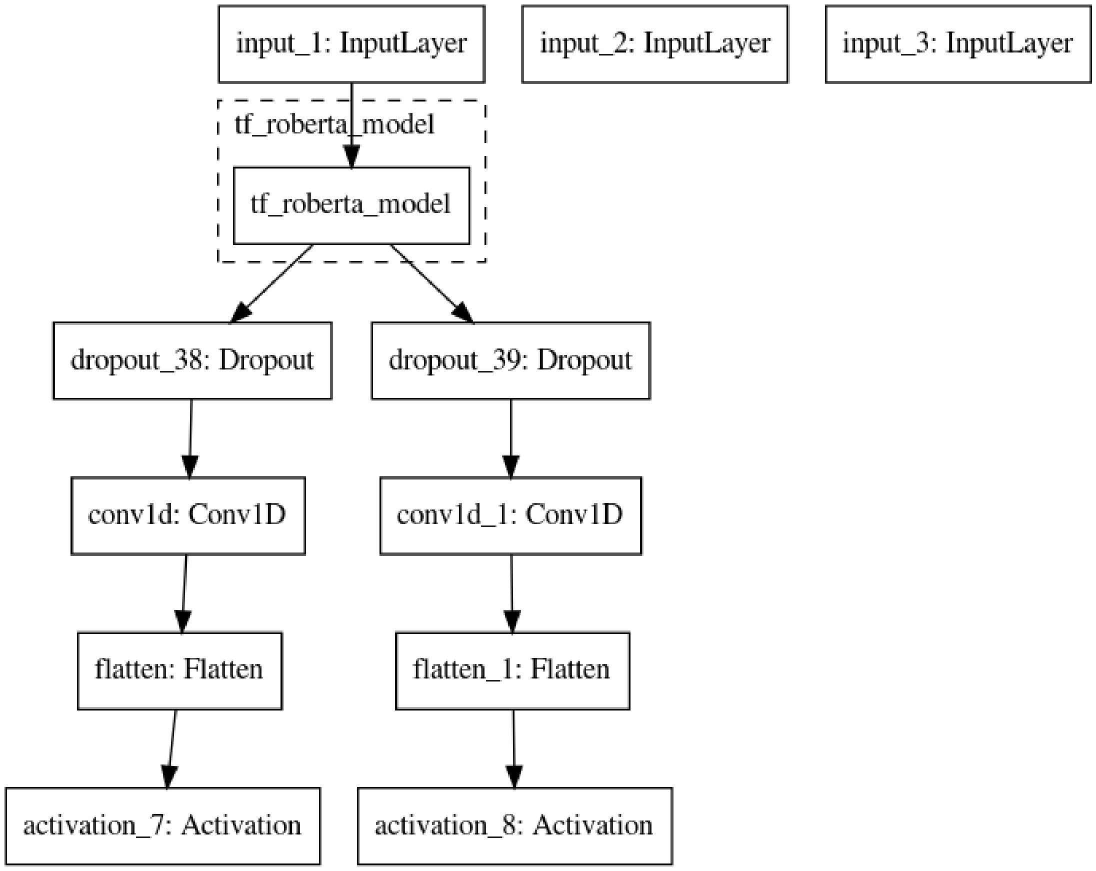


Fig. 8 The above illustration denotes the RoBERTa base with a custom question answering head.

The three input layers are:

- Input IDs
- Attention Masks
- Token Type IDs

The 2 output layers are:

- Start tokens
- End tokens

I make use of stratified cross-validation with 5 folds, thereby training 5 RoBERTa models. The start/end token predictions of these 5 models are averaged to get the final start/token indices.

Data Inference

Once the model is trained and the weights are saved, I apply the same preprocessing and transformation steps on the test dataset and then I pass the transformed data for model prediction. I ran the inference notebook “tweet-extraction-inference” on Kaggle to get my score on the leaderboard for the test data (as we don’t have true labels for test data).

Refinement

The refinement process was carried out by fixing random seed wherever possible.

My initial model did not make use of stratified cross-validation and had a train/test split of 80/20. I also had a TOKEN_MAX_LEN = 96. Thus, my final predictions were made from just one model. This initial model scored 0.705 on the Kaggle leaderboard.

I then changed the TOKEN_MAX_LEN to 148 for the reason explained in the data pre-processing section. I also made use of stratified cross-validation with 5 folds. Thus, I trained 5 RoBERTa models on the 5 folds, and the predictions are made by averaging the results of the 5 models. This improved my test score on the Kaggle leaderboard to 0.707.

4. Results

Model Evaluation and Validation

The final architecture and hyperparameters were chosen because they performed the best among the tried combinations. The final architecture is described in Figure 8 and these are the hyperparameters on which it was trained:

- ❖ The Dropout value is 0.1
- ❖ The Convolution layers have 1 filter and a kernel size of 1.
- ❖ The Convolution layer have a stride of 1, so the shape of the output is (TOKEN_MAX_LEN, 1)
- ❖ The optimizer is Adam with a learning rate of 3e-5
- ❖ The loss function is categorical cross-entropy.
- ❖ The training for each of the 5 fold runs for 3 epochs.

To verify the robustness of the final model, an out of sample set was used for validation. The out of sample set was generated during each fold of the stratified cross-validation. As we observed during EDA that our test set has the same distribution as our training set, we can be sure that the test score will be around the same as our validation score.

The validation Jaccard score is 0.7066 and the test Jaccard score from the Kaggle leaderboard is 0.707.

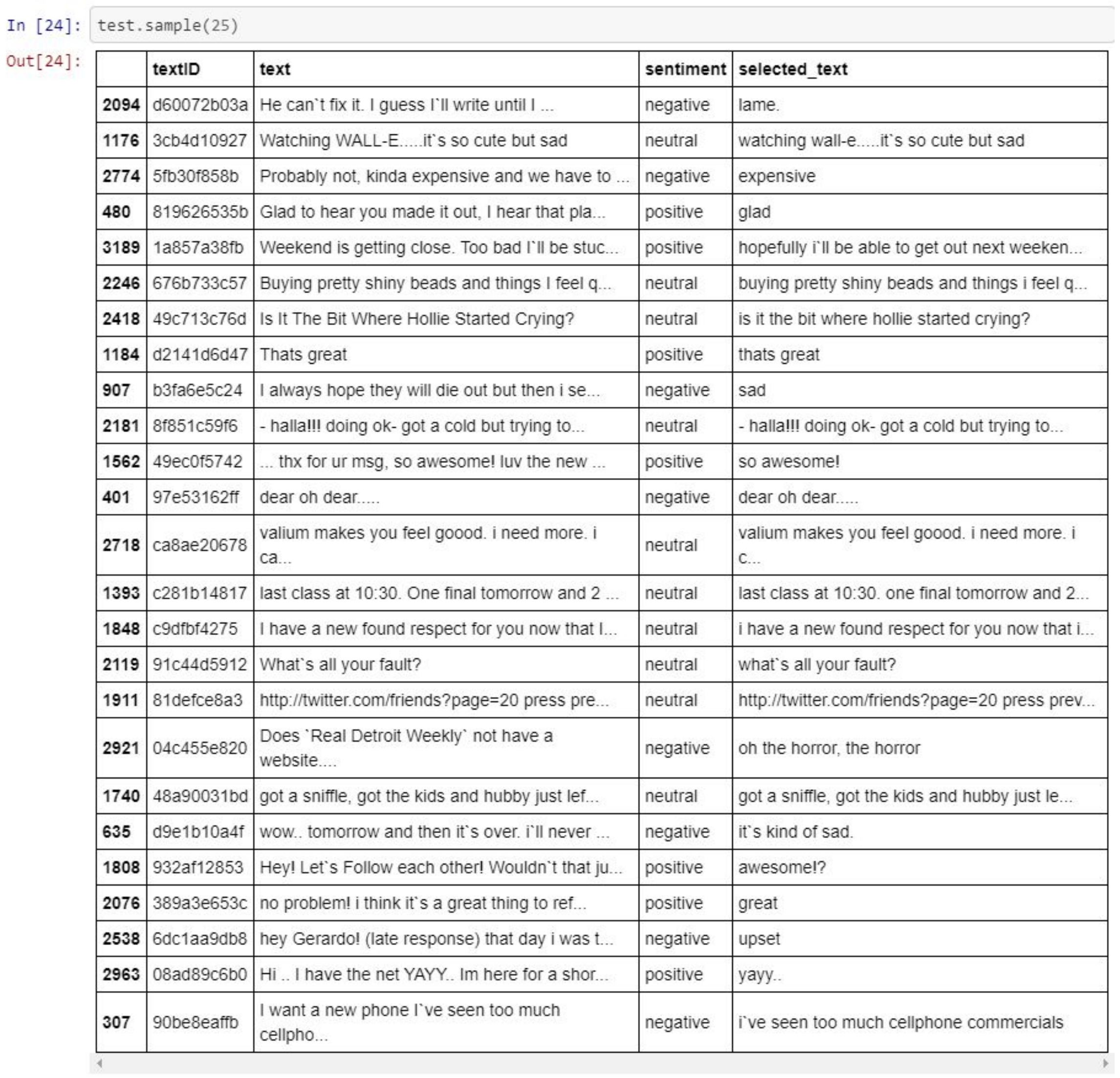
Justification

The benchmark score was 0.651 and the RoBERTa model score is 0.707. As the test dataset does not have any true labels, we can be sure that the score observed on the test set is robust.

The improvement with the RoBERTa model is statistically significant and the model generates good predictions to offer explainability (selected_text) to a tweet's sentiment.

5. Conclusion

Free-Form Visualization



```
In [24]: test.sample(25)
Out[24]:
```

	textID	text	sentiment	selected_text
2094	d60072b03a	He can`t fix it. I guess I`ll write until I ...	negative	lame.
1176	3cb4d10927	Watching WALL-E.....it`s so cute but sad	neutral	watching wall-e.....it`s so cute but sad
2774	5fb30f858b	Probably not, kinda expensive and we have to ...	negative	expensive
480	819626535b	Glad to hear you made it out, I hear that pla...	positive	glad
3189	1a857a38fb	Weekend is getting close. Too bad I`ll be stuc...	positive	hopefully i`ll be able to get out next weeken...
2246	676b733c57	Buying pretty shiny beads and things I feel q...	neutral	buying pretty shiny beads and things i feel q...
2418	49c713c76d	Is It The Bit Where Hollie Started Crying?	neutral	is it the bit where hollie started crying?
1184	d2141d6d47	Thats great	positive	thats great
907	b3fa6e5c24	I always hope they will die out but then i se...	negative	sad
2181	8f851c59f6	- halla!!! doing ok- got a cold but trying to...	neutral	- halla!!! doing ok- got a cold but trying to...
1562	49ec0f5742	... thx for ur msg, so awesome! luv the new ...	positive	so awesome!
401	97e53162ff	dear oh dear.....	negative	dear oh dear.....
2718	ca8ae20678	valium makes you feel goood. i need more. i ca...	neutral	valium makes you feel goood. i need more. i c...
1393	c281b14817	last class at 10:30. One final tomorrow and 2 ...	neutral	last class at 10:30. one final tomorrow and 2...
1848	c9dfbf4275	I have a new found respect for you now that I...	neutral	i have a new found respect for you now that i...
2119	91c44d5912	What`s all your fault?	neutral	what`s all your fault?
1911	81defce8a3	http://twitter.com/friends?page=20 press pre...	neutral	http://twitter.com/friends?page=20 press prev...
2921	04c455e820	Does `Real Detroit Weekly` not have a website....	negative	oh the horror, the horror
1740	48a90031bd	got a sniffle, got the kids and hubby just lef...	neutral	got a sniffle, got the kids and hubby just le...
635	d9e1b10a4f	wow.. tomorrow and then it`s over. i`ll never ...	negative	it`s kind of sad.
1808	932af12853	Hey! Let`s Follow each other! Wouldn`t that ju...	positive	awesome!?
2076	389a3e653c	no problem! i think it`s a great thing to ref...	positive	great
2538	6dc1aa9db8	hey Gerardo! (late response) that day i was t...	negative	upset
2963	08ad89c6b0	Hi .. I have the net YAYY.. Im here for a shor...	positive	yayy..
307	90be8eaffb	I want a new phone I`ve seen too much cellpho...	negative	i`ve seen too much cellphone commercials

Fig. 9 Examples of the 'selected_text' predicted by the RoBERTa model on the unseen test dataset.

Reflection

The process used for this project can be summarized using the following steps:

1. An initial problem and relevant, public datasets were found
2. Exploratory data analysis was performed on the dataset to gather a better understanding of the data
3. The data was pre-processed and transformed as per the specification of the RoBERTa model
4. The RoBERTa base model was paired with a custom question answering head
5. Stratified K fold cross-validation was used to train 5 models and validation was performed on out of sample sets
6. The inference was done in a separate notebook and submitted on Kaggle to get a score on the test dataset

Steps 3 and 4 were the most difficult. This was my first time using a transformer network and thus, I had to do a lot of reading on the tokenization techniques required by the transformer networks and different transformer algorithms.

The most interesting aspect of this project is the real-world exposure that this provides as sentiment analysis is common at the workplace and I had already deployed 2 sentiment analysis projects in my professional capacity. This project can be a good add-on with an existing sentiment analysis pipeline, where one could extract words that may be responsible for a given sentiment.

Improvement

Right now, this implementation does not perform very well when tweets have Unicode characters. This limitation was due to the Huggingface tokenizer not decoding Unicode characters. To improve this, a different implementation of a byte-level BPE tokenizer can be used or a different transformer network can be explored.

Another area of improvement can be changing the transformer algorithm to something like XLM-RoBERTa. I encountered this while reading the documentation of RoBERTa on the Huggingface transformer library. I've tried to make my code resilient to changes in tokenizer and algorithms, and I believe this can be substituted with minimal efforts.

6. Citations

[1] Agarwal, Apoorv, et al. "Sentiment analysis of twitter data." Proceedings of the Workshop on Language in Social Media (LSM 2011). 2011.

[2] Pak, Alexander, and Patrick Paroubek. "Twitter as a corpus for sentiment analysis and opinion mining." LREc. Vol. 10. No. 2010. 2010.

[3] Saif, Hassan, Yulan He, and Harith Alani. "Semantic sentiment analysis of twitter." International semantic web conference. Springer, Berlin, Heidelberg, 2012.

[4] Rosenthal, Sara, Noura Farra, and Preslav Nakov. "SemEval-2017 task 4: Sentiment analysis in Twitter." arXiv preprint arXiv:1912.00741 (2019).

[5] “Tweet Sentiment Extraction.” Kaggle, www.kaggle.com/c/tweet-sentiment-extraction/overview/description.

[6] Nkoprowicz. “A Simple Solution Using Only Word Counts.” Kaggle, Kaggle, 7 Apr. 2020, www.kaggle.com/nkoprowicz/a-simple-solution-using-only-word-counts.

[7] Huggingface. “Huggingface/Tokenizers.” GitHub, 24 Apr. 2020, github.com/huggingface/tokenizers.

[8] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

[9] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[10] Cdeotte. “TensorFlow RoBERTa - [0.705].” Kaggle, Kaggle, 14 Apr. 2020, www.kaggle.com/cdeotte/tensorflow-roberta-0-705/notebook#Build-roBERTa-Model.