Tweet Sentiment Extraction

Ajeet Kumar Dwivedi
ECE Department
Western University
London, Canada
adwived3@uwo.ca

Vamsi Pasupuleti
ECE Department
Western University
London, Canada
vpasupul@uwo.ca

***Abstract*— The Cyber World has become a rudimentary requisite for everyone with the utilization of Web and internet. With the growth of web, volume of data associated with it has also grown prodigiously. Concretely, social media has evolved as a major platform for sharing opinions, thoughts and ideas. This platform is expeditiously gaining popularity as it facilitates people to express their sentiment, views, opinions and ideas. Sentiment Analysis is one of the ways to classify the responses of the users and their sentiment behind any message but knowing only the sentiment is not enough these days. As companies are transcending the boundaries to serve their customer in best possible way, meaningful extraction of the customer language behind any review for the given service can help both the parties' customer and business to grow. This project solves this problem and aims to deliver the words or phrases which are responsible for the categorization of tweets in different sentiment categories (positive, negative, neutral). Different AI based model results shows that it is quite achievable to predict the combination of words responsible for the sentiment. Since it's a Natural language processing (NLP) problem which require different techniques of pre-processing, Bidirectional Encoder Representations from Transformers (BERT) model with jaccard score of around 63% has proved as one of the best learner and predictor, followed by ensemble models Xgboost with score of 62% and bi-directional Gated Recurrent Unit (GRU) with jaccard score 55%.**

Keywords—BERT, GRU, Xgboost, ensemble, NLP

I. INTRODUCTION

Sentiment Analysis is the finding of different sentiments positive, negative and neutral using machine learning model and NLP techniques. This process consists of identification, classification and extractions of various tweets or reviews. Generally, 1 being given to positive -1 for negative and 0 for neutral tweets [11]. Contrarily, Sentiment Analysis Extraction involves sentiment analysis with text data analysis and then determining the words or language resulting in different sentiments [11][8]. It can be utilized in numerous industries and use cases such as, Client benefit, Brand Checking or political campaigns but one of the most excellent use cases is Social Media Content Monitoring. Using customer feedback, customer surveys, product reviews and social media discussions businesses can gain a better understanding of their clients' opinions, which is becoming ever more important in order to satisfy their needs [12][14]. Having the capacity to capture sentiment in dialect is imperative in these times where choices and responses are made and upgraded right away. So, what about those words which reflects the opinion and views of the user. Subsequently, we choose out the portion of the tweet (words or phares) that best reflects the sentiment [5][8]. Social media Users create an enormous amount of content, which is arduous to analyze by traditional software or coding. Therefore, these contents must be automated using Machine learning models.

Decision of individuals or new users are influenced by the online reviews and feedback provided by other users. From buying the product or purchasing the service, it has become a practice to first check the online reviews then make a decision. Business revenues depends a lot on customer retention and new customer acquisition [5][9]. Hence businesses are trying to know the reason why their customers are happy, sad or neutral. Ascertaining the reason behind the customer reviews can help company to directly categorize the users in more than three categories such as 'delighted', 'happy' from positive perspective, 'awful', 'bad' from negative perspective and content and okay for neutral [7].

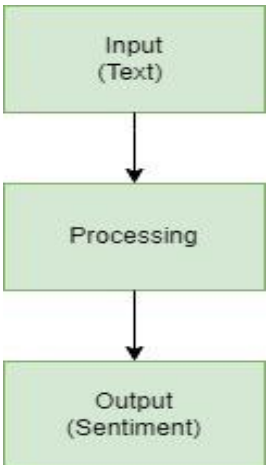
Less number of people use to write the 100-120 words for any statement, but in reality, these are the rare customers who are either most happy and optimistic or distressed and pessimistic from the businesses [1][14]. Capturing their language behind the emotions can help businesses ameliorate on their service and product. Moreover, these customers reviews define the new customer addition to the businesses.

Majorly used language for sentiment analysis is English but there are thousands of languages where users can give their opinion, analysing the multilingual text or word behind sentiments can provide enormous meaningful [1][11]. Furthermore, finding the most frequent word behind any sentiment for example happy for positive, sad for negative and okay for neutral conveniently help model to learn faster structured way.

II. MOTIVATION

The sum of information accessible on the internet for web clients is colossal. Thousands of social media discourses, client surveys, and overviews are nearly inconceivable to manually sort, so we ought to use machine learning model to analyse the information and performs the specified operations. We are given with a few contents' information at the side their positive, negative and unbiased opinion and we got to discover phrases/words that best bolster the opinion [4]

Sentiment Analysis Extraction works on the top of Sentiment analysis, below diagram shows the difference between them in terms of the process and flow.[1]



Twitter Sentiment Extraction GitHub Link: https://github.com/Vamsikrishnapasupuleti/MachineLearning

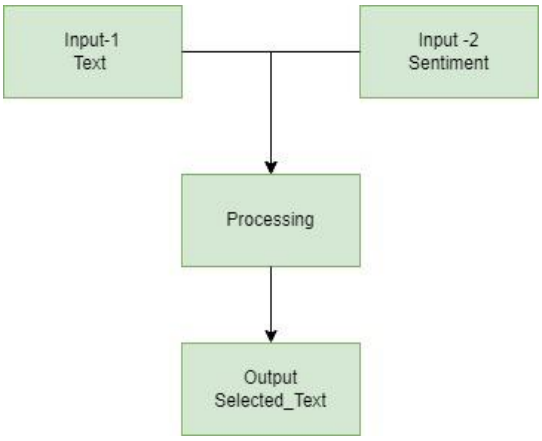


Fig:2. Diagram showing the flow of tweet Sentiment Extraction

From the given charts its translucent that Tweet Sentiment Extraction works with two inputs (text, Sentiment) and provides the selected text as an output. For example, if there is a sentence that 'I am soooo happy that I will be graduating this term' here 'soooo happy' is the selected text which exemplifying that it is a positive sentiment. 'I am very sad that I won't be the part of western university after this term'. In this sentence 'very sad' is the word which is responsible for the negative sentiment.

III. Related Work

Each single human activity depends on their believes, views and opinions as these are powerful components in our decision-making. In regard to twitter Sentiment Analysis, we can find many research paper, however for Twitter Sentiment Extraction limited research has been proposed. Chiranjeev et al. [1] proposed a solution where they achieved 73.12% score using DistilBERT model with hypertuning and fine tuning, however earlier it was 69% only. They also implemented BERT model but was getting jaccard score somewhere around 64.5%. Later they implemented Facebooks RoBERTa with score of 67.98 and Google's ALBERT with 62.39% score. All their result is based on the Jaccard score which is formulated on the concept of intersection and union. Shriram s [2] proposed and worked on different models. Started with deep learning base Recurrent Neural Network (RNN) Model then used Long short-term memory (LSTM) and Gated Recurrent Unit (GRU), both these models are taking the text and sentiment as input and gives selected text as output, but he ended up only with 60% score. Later he implemented bi-directional GRU but not much help in score. After switching to Bidirectional Encoder Representations from Transformers (BERT) model. He implemented Robustly optimized BERT (RoBERTA) tokenizer model and got the best accuracy jaccard score around 70%. He is flattening both input layer and then concatenating it for the next layer. Dipanshu Rana [3] implemented Deep learning model such as Long short-term memory (LSTM) with score 57% only. Later he implemented bi-directional LSTM and got the Jaccard score somewhat higher, around 63%. He used two input layers and without flattening, he is concatenating the layer and with simple approach forwarding this layer to the next layer. Agarwal et al. [12] has proposed a solution using fixed indicator based on Part of Speech POS. The expressions extricated are not whole expressions and are instep as it were a bunch of one or two words only, it also has a limitation of not performing well for phrases rich in sentiment.

IV. Dataset

Kaggle is currently running a competition with this dataset. In this competition, phrases were drawn from Figure Eight's Data for Everyone platform [14]. This Dataset consists of two data files train.csv and test.csv, where records present in training data are 27481 while test data contains 3534 rows. Columns present in the dataset include:

'textID': signifies unique id for each row of data.
'text': column contains the text data of the tweet which is the actual language or sentence used for review.
'sentiment': sentiment behind the selected language (positive/negative/neutral).
'selected_text': phrases /words from the text column that best describes the sentiment behind writing the tweet [4].

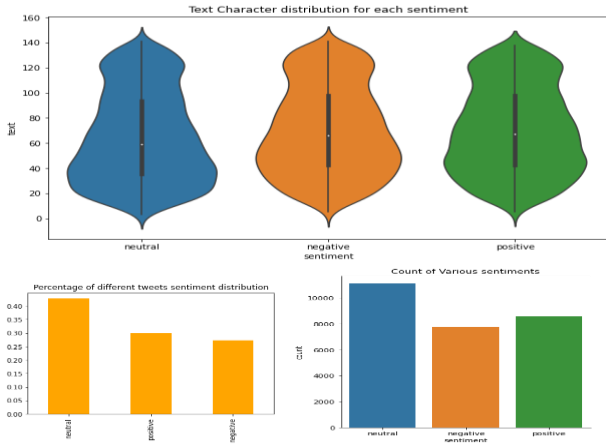
A tweet and its sentiment(label) are displayed in each row. The training set contains a word or phrase derived from a tweet ('selected_text') that captures the given sentiment. In order to avoid misrepresenting the CSV, we will remove the beginning and ending quotation marks from the text field while parsing the CSV file. Also, we will be considering and showing some characters falling within the word/phrases span such as commas, starts, spaces, etc.

V. Methodology

A. *Exploratory Data Analysis*

We have performed the exploratory data analysis on sentence length which is number of characters in the sentence including the special character. Furthermore, we analyzed the words counts in that sentence.
EDA for sentence length is:

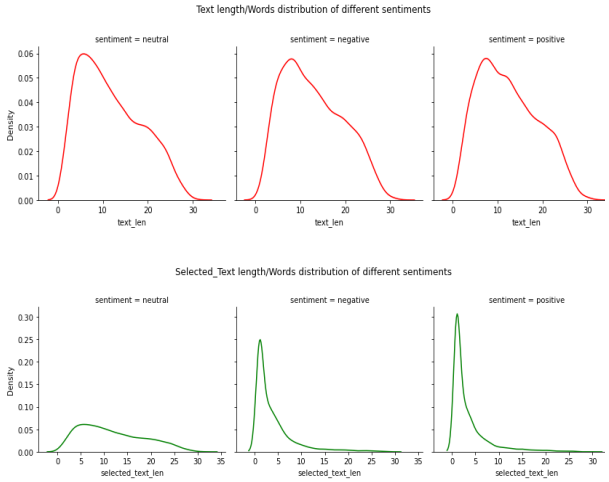
- Of the total tweets, about 40% of the tweets are neutral followed by positive 31% and negative 28%.
- Total No. of tweets are around 27k in which neutral comprises 11k, positive tweets are 8.5k and negative tweets are approx. 7.7k.
- Maximum length of the sentence is 138 chars while min is 2 char.
- Most of the tweets are lying in range between 35 to 100 characters.
- Neutral tweets: range from [35 to 95], Positive tweets: range from [38 to 98] and Positive tweets: ranges from [40 to 100]



Twitter Sentiment Extraction GitHub Link: https://github.com/Vamsikrishnapasupuleti/MachineLearning

EDA for word count is:

- Majority of the tweets text word fall in range 15 to 25 for all sentiment categories.
- For all sentiment categories, only few tweets of text word counts greater than 30 or less than 5.
- Maximum words for the longest tweet is 33, while minimum is 2.



B. Preprocessing

The Preprocessing includes two steps,

1. Data Cleanup
2. Data Preparation

Data Cleanup:

- Strip the spaces
- Lower the text
- Remove all punctuations except '*' & ' ' ' (RegEx: [^\w\s*\']+)
- Remove URL's starting with http or https (RegEx: \S*https?:\S*)

We are stripping of the spaces and lowering the text to generalize the content.

We are not performing lemmatization (not converting verb form to noun), Stemming(not removing prefix or suffix from words like es…) and not removing stop words because the problem statement addressing here is to predict the phrase that best describes the sentiment but not the sentiment.
Hence we need the sentence to be in its original form but generalized like removing the punctuations except '*' & ' ' ', removing the URL's. We need '*' because that describes the abusive words, and it has significant importance in negative sentiment tweets.[25]

Example : The quick brown fox jumps over the lazy dog***** , he's quick http://xyz.com ###.
Result: the quick brown fox jumps over lazy dog***** he's quick

Packages Used: nltk, re, string

Data Preparation:
The Data Preparation is divided into 3 parts,

1. Using Doc2Vec model (XGB)
2. Using GloVe (GRU)
3. Using Bert (BERT)

Word Embeddings:

Before discussing about the word embedding models let us know about word embedding. Word embedding is nothing but converting a text into continuous values in the form of a vector so that model can understand the numbers and perform computations and produce results. Machines does not understand the text data and hence we are performing word embedding step.

Doc2Vec & GloVe:

Before selecting Doc2Vec model we did a thorough analysis on available word embedding models present in the industry.

Initial models are Bag of Words (BoW) & TDIDF vector models. These models work on basis of one hot encoding referring to the dictionary of words which were given initially. The disadvantage of these models are High dimensionality and sparse matrix (as a word in index 5000 of a dictionary 10000 will be represented as all 0's except 5000th index as 1 which means the matrix has 10000 dimensions and only has one 1 and rest are 0's which refers to sparse matrix i.e., dominant 0's which refers to imbalance which can affect the model). Also, these models neither capture the correlation between the words nor the context.

To better address these issues Word2Vec model was designed. This model does not generate word vectors based on the number of words in dictionary, but does it based of the feature representation.(a word male is related to gender and hence generates a value like 0.9 but other word like this is not related to gender at all and assigns a value of 0.1). Word2Vec model generates features and assigns values based on correlation.

GloVe refers to Global Vectors for word representations. This algorithm is developed by Stanford. Both models assign only one unique vector for each word but the difference between GloVe & Word2Vec is GloVe considers the co-occurrence of words in global context, but Word2Vec does not. It is not always said that GloVe is better than Word2Vec, but the performance depends on type of data we choose.[23]

Doc2Vec is easy to use when we are dealing with the sentences. Since the data we are working with has sentences, if we choose Word2Vec then for each word vector obtained from the model we then need to average the vectors for each word in a sentence and then add 0 vectors for rest of the max length we choose at the beginning. To avoid this processing, We, used Doc2Vec model which infers vectors based on the sentence.

Both models discard the new words which is a disadvantage. Hence we trained the models on WIKI corpus data which is large data with Wikipedia content and has less chances of unknown words. [24]

Example:

```
test_data = word_tokenize("I love Machines".lower())
v1 = modeld2v.infer_vector(test_data)
```

Result: [-0.00823853 0.0125465 -0.05362745 -0.00804744 -0.03190618 -0.00868102 -0.04342219 0.04068456 0.02885487 -0.00139011 -0.0370013 0.01534083 -0.0025305 -0.05201721 0.03253455 -0.0085752 0.02434799 -0.07629792 -0.01320806 0.05194129 -0.03206015 -0.0479126 0.00537943 0.00213766 -0.02185634 -0.09854241 0.00201005 0.02742022 -0.0083191 0.04130479 0.06971811 -0.0339016 0.01470919 -0.04786061 0.0945525 -0.01840335 0.03125513 -0.02065532 -0.02845261 0.0605186]

Note: The above example generates 40 dimensions as we have specified the vec_size parameter as 40.

BERT Tokenizer:

Bert model inputs are different from other models. Bert model takes input ids and attention masks and then generates the word embedding vectors. Input ids are nothing, but each word will be assigned to a number based on the context and position. Attention mask is, if we set a max length for bert tokenizer then all sentences which are less than max length will be assigned 0's for input ids but attention mask tells us what are the exact ids which are related to original sentence i.e., all input ids which are related to original sentence is represented as 1 and all other special tokens and padded sequences are represented as 0.

Bert Tokenizer has some special tokens like CLS ,SEP, UNK which represents start of sentence , end of sentence and unknown words in a sentence respectively. Hence Bert has inbuilt capability to handle unknown words thus by not loosing the order of sentence which helps better in prediction.

Bert generates word embeddings based on the context .i.e., a word can multiple vector representations based on the context and position of the word and hence Bert acquires supremacy in NLP tasks.[7]

We used bert-base-uncased pretrained tokenizer to generate the input ids, attention masks and offsets.

```
Hello, How r u??
Input Id's:
 [101, 7592, 1010, 2129, 1054, 1057, 1029, 1029, 102]
Tokens:
  ['[CLS]', 'hello', ',', 'how', 'r', 'u', '?', '?', '[SEP]']
Offsets:
 [(0, 0), (0, 5), (5, 6), (7, 10), (11, 12), (13, 14), (14, 15), (15, 16), (0, 0)]
```

Packages used: genism, transformers

Sentiment encoding:

We are using sentiment feature for our model as input and hence we are converting the sentiments neutral, positive, and negative using binary encoding .[15]

Binary encoder works on binary representation. If outputs are 3 then the binary representation will be 01,10,11 (i.e., 1,2,3 in binary representation). Here our values are 3 and hence the values will be converted to two columns with values of 01 for neutral, 10 for positive & 11 for negative.

Packages used: category_encoders

C. Models

1. XGB
2. BERT
3. Bi-directional GRU

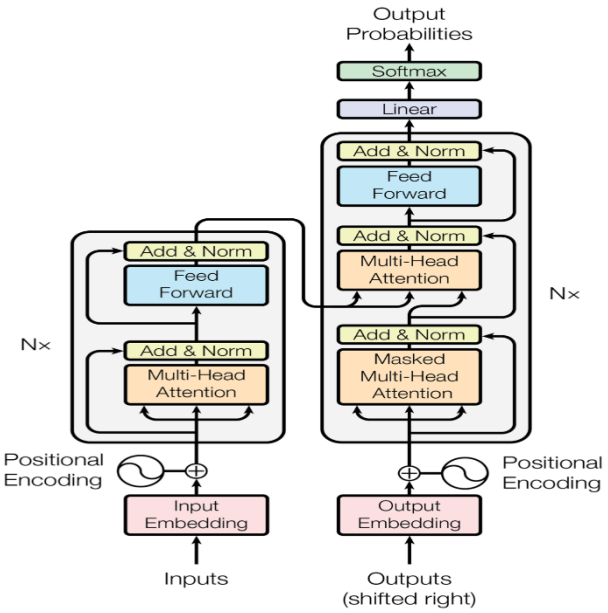
XGB:

XGB is ensemble boosting technique .It is a sequential model where it does adaptive error optimization based on previous errors (i.e., in errors in previous tree is penalized to obtain best accuracy).

We are using XGBRegressor, a wrapper class for sklearn. Since model should predict start & end indices i.e., multi output prediction, XGB is not compatible of generating multi output . Hence we used MultiOutputRegressor from sklearn which will make model produce the desired number of outputs.

Since the outputs are continuous values we used mean squared error as loss function .

BERT:



Bert is Bidirectional Encoder Representation for Transformers. Transformers is a combination of both Encoder and decoder. The Encoder takes in the input ids and attention mask and produce the word embeddings for the inputs we passed. This is then passed to decoder as input along with the other input to decode and then get the translated output. (For Example, language translation We pass input ids and mask of English as input to encoder where it learns English, and grammar and the output of encoder is then passed as input to Decoder along with the other language input ids and mask to decode and provide the translated output). Stacking these Encoder is known as Bidirectional Encoder Representation for Transformers(BERT). [21]We used bert-base-uncased model which has 12 layers, 768 hidden units, 12 attention heads and 110M parameters. This model is pretrained model and then we are fine tuning the weights in the layers with our inputs during model training.[20]

Bert is faster because words will be processed simultaneously, and it can learn from both directions which makes it bidirectional, and context can be better understood in both directions and hence the results are best.

We just need to connect the output of Bert to a desired output layer to obtain the required predictions. We have also added one hidden layer which has improved accuracy when compared to Bert's output directly connected to output layer.

The outputs we got in final output layer is array of 0's and 1's and hence we used cross entropy as loss function for this model.

GRU(Gated Recurrent Units):

GRU has less computational complexity when compared to LSTM because it has only two gates, update & reset.

Bidirectional GRU has capability of processing from both forward & backward directions in every time step and can produce better results when compared to Unidirectional GRU. Hence It understands better context in natural language processing tasks.

We used two input layers one for text and other for sentiment and concatenating the input layers and the passing through bidirectional GRU layer. We have also used batch normalization layer to normalize the outputs of previous layer to achieve mean to 0 and standard deviation to 1 before feeding it as inputs to next layer which will increase stability of model. The output layer has 2 outputs since we are predicting the start and end indices.

We didn't use any hyperparameter tuning techniques for GRU and did a manual search on few parameters and got the best of it.

D. Hyperparameter Tuning:

XGB:

For tuning XGB we chose TuneSearchCV which is part of ray tune library which works on distributed Hyperparameter tuning . It is 3x times faster than the exhaustive GridSearch and more efficient than Random Search model since it selects parameters based on Bayesian optimization search algorithm. It has a mechanism for EarlyStopping & can reduce time complexity. We gave EarlyStopping parameter as Median Stopping Rule. The Median Stopping Rules stops the trail if the results are less than the median of other trails. Below are the parameters we tuned and best parameters obtained after tuning. We are not taking same set of records for every tree we are subsampling using subsample parameter and the best we achieved is 0.6.[22]

Parameters:

```
xgb_param_grid = {
    'estimator__max_depth': [10, 11, 12],
    'estimator__n_estimators': [200, 500, 1000],
    'estimator__learning_rate': [0.1, 0.01, 0.001, 0.004],
    'estimator__random_state': [1],
    'estimator__subsample':[0.7, 0.8, 0.5, 0.6]
    'estimator__tree_method':['gpu_hist']
  }
```

Best params:

```
learning_rate=0.004,
max_depth=10,
n_estimators=1000,
subsample=0.6
```

BERT:

We used optuna for BERT hyperparameter tuning. Tuning is done on batch_size, learning_rate, optimizer, hidden_units. Optuna creates a study object and studies the objective functions for n trails. If objective function return loss then specify direction as minimize else if objective functions return accuracy make it maximize instead. We are using pruner parameter to discard badly performing trails. We used MedianPruner which discards the trails if the objective falls below the median of other trails at similar points in time. We can also define our own Pruning functionality.[16]

Optuna does not have feature to save the model it will just return the best trail values. Hence we designed a custom callback function and saving the model in the best trial. Also, we are using Layerwise Adaptive Momentum for large batch trainings (LAMB) optimizer as it is more effective in terms of BERT as it works based on Adam optimizer but adjusts Adam's learning rate more accurately based on the trust ratio (magnitude of gradients to weight decay)

Bert:

Hyperparameter tuning → Optuna

```
params = {
        'batch_size': trial.suggest_int("batch_size", 8, 64, step=8),
        'learning_rate': trial.suggest_loguniform('learning_rate', 3e-5, 1e-3),
        'optimizer': trial.suggest_categorical("optimizer", ["AdamW", 'Lamb' , 'Adam'
, SGD']),
        'hidden_units': trial.suggest_int("hidden_units", 64, 128, step=32)
    }
```

Best Parameters: batch_size =32, learning_rate = 3e-05, optimizer = Lamb, hidden_units = 128

VI. RESULT AND ANALYSIS

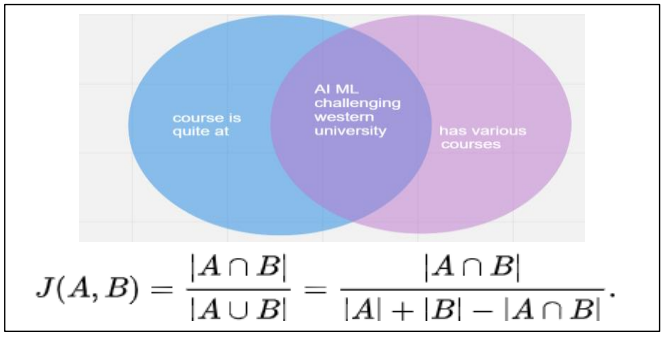
A. Evaluated Metrics

The Jaccard Score or Jaccard Similarity is one of the statistics used in understanding the similarity between two sets.

Sentence 1: AI ML course is quite challenging at western university

Sentence 2: western university has various challenging AI ML courses

J (A, B) = 5/ (9+8-5) = 0.41



B. Benchmarking

We first used bidirectional GRU as our base model and achieved a jaccard score of 58%.

Then we did a thorough research and used bert model which outperforms other models in natural language processing tasks and achieved jaccard score of 63.9% after hyperparameter tuning.

Twitter Sentiment Extraction GitHub Link: https://github.com/Vamsikrishnapasupuleti/MachineLearning

Since preprocessing-1 is a kind of normal regression problem where we converted all the text to vectors of 40 dimensions and sentiment to binary encoder technique & predicting start and end indices which are continuous values . XGB is latest model which works better for this kind of data we used that & did hyperparameter tuning and got a jaccard score of 61.089%

We didn't use sentiment in Bert model and achieved a score of 63.9% but if we use sentiment also we will get more jaccard score.

C. Results:

Model	Hyperparameters	Jaccard score	Time Complexity (with GPU)
BERT	batch_size=32 learning_rate=3e-05 optimizer=Lamb hidden_units=128	63.934	2hr 50 min
XGB	learning_rate=0.004 max_depth=10 n_estimators=1000 subsample=0.6	61.089	4hr 30min
Bi-directional GRU	kernel_initializer=glorot_uniform epochs=80 batch_size=256 validation_size=64	58.667	2hr 10min

VII. CONCLUSION

Since pre- processing and post-processing both approaches attempt to generalize input and output, they can produce results that depart from the actual truth that model anticipated in the first place. As a result, the total accuracy is reduced, and the model's accuracy falls short of what it might have attained.

Bert has the achieved the best accuracy with a score of 63.9% followed by XGB and then GRU. Even though XGB has got better accuracy than GRU (without tuning). Predictions on GRU will perform better than XGB for unseen data as XGB works on rectifying the previous errors by penalizing the errors but GRU understands context in both directions. Also ,with hyperparameter tuning GRU can achieve better score than XGB.

Future task:

We have saved the best model from bert, and we are trying to design a web preview where user enter text and sentiment as inputs, and we process the using python code and get the text that best describes the sentiment and show it on the user screen using express js as frontend and python as backend and hosting the application on cloud using Heroku.

Also , using sentiment for bert model will boost the jaccard score near to 70% which will be nearer maximum score achieved for this competition i.e., 73%

We will also try to implement with bert-large-uncased model which has larger parameters and more layers which can give better score than the base model.

Facebook's Roberta model has better has better performance when compared to bert model. Hence we can achieve the best score after implement the above changes to current model.

GitHub Link:
https://github.com/Vamsikrishnapasupuleti/MachineLearning

REFERENCES

[1] Tweet Sentiment Extraction Chiranjeev Mishra, Eshit Bansal and A. Helen Victoria, January 5, 2022

[2] Shriram s "Tweet Sentiment Extraction" Nov 2, 2020

[3] Dipanshu Rana, "Tweet Sentiment Extraction" Nov 3, 2021

[4] https://www.kaggle.com/c/tweet-sentiment-extraction

[5] Vishal A. Kharde, Department of Computer E, "Sentiment Analysis of Twitter Data: A Survey of Techniques," Volume 139 – No.11, April 2016.

[6] Abdullah Alsaeedi1[el], "A Study on Sentiment Analysis Techniques of Twitter Data", No. 2, 2019.

[7] https://huggingface.co/bert-base-uncased

[8] https://www.kaggle.com/vpkprasanna/eda-analysis-wordcloud-bert-classification

[9] https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-bert/

[10] https://medium.com/distributed-computing-with-ray/hyperparameter-optimization-for- transformers-a-guide-c4e32c6c989b

[11] K. Dashtipour, S. Poria, A. Hussain, E. Cambria, A. Y. Hawalah, A. Gelbukh, and Q. Zhou, "Multilingual sentiment analysis: State of the art and independent comparison of techniques," Cognitive Computation, p. 757–771, 2016.

[12] B. Agarwal, V. K. Sharma, and N. Mittal, "Sentiment classification of review documents using phrase patterns," in 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2013, pp. 1577–1580.

[13] P. M. Vu, H. V. Pham, T. T. Nguyen, and T. T. Nguyen, "Phrase-based extraction of user opinions in mobile app reviews," in 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), 2016, pp. 726–731

[14] https://appen.com/datasets-resource-center/

[15] https://contrib.scikit-learn.org/category_encoders/

[16] https://optuna.readthedocs.io/en/stable/

[17] https://towardsdatascience.com/5x-faster-scikit-learn-parameter-tuning-in-5-lines-of-code-be6bdd21833c

[18] https://github.com/abhishekkrthakur/bert-sentiment

[19] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[20] https://jalammar.github.io/illustrated-bert/

[21] https://machinelearningmastery.com/the-transformer-model/

[22] https://docs.ray.io/en/latest/tune/api_docs/sklearn.html

[23] https://radimrehurek.com/gensim/models/doc2vec.html

[24] https://markroxor.github.io/gensim/static/notebooks/doc2vec-wikipedia.html

[25] https://docs.python.org/3/library/re.html

CONTRIBUTION:

Ajeet → Code: GRU, EDA
Vamsi → Code: BERT, Data Preprocessing
Ajeet & Vamsi → XGB