StumbleUpon Evergreen Classification Challenge

- By Satyam Kumar Yadav

Email - satyam.kryadav.che19@iitbhu.ac.in
GitHub - Satyam-kumar-yadav

Project Abstract

In this project, I have demonstrated how I have tackled the StumbleUpon Challenge and what are the problems I have faced during the challenge.
I tried different lengths of words and used the Pretrained model from the Hugging face library for the text classification. I used my previous notebook work in which I got the gold medal and 1300+ views. I have used Google Colab for completing the task.

Project Description

I may have used models like Tf-IDF vectorizer with Logistic regression or using word embeddings layer with LSTM and GRU's (i have used it in my previous notebook Example) but then I decided why not use the bert tokenizer and bert model for the long texts and I found it quite interesting and learned a lot from it.

1 - Loading tsv files using pandas which I don't know. #1
2 - Explored the data and found relevant things to get started such as which part can help me.
3 - Then I created the new DataFrame from (boilerplate and label) column of the Dataset.

So Now come the fun part of using Bert😁😁

1 - First the DataSet needs to be processed (usually not needed for the bert).
(The reason I processed the data because since I can't use max length > 512, so preprocessing can help to get important words from text)

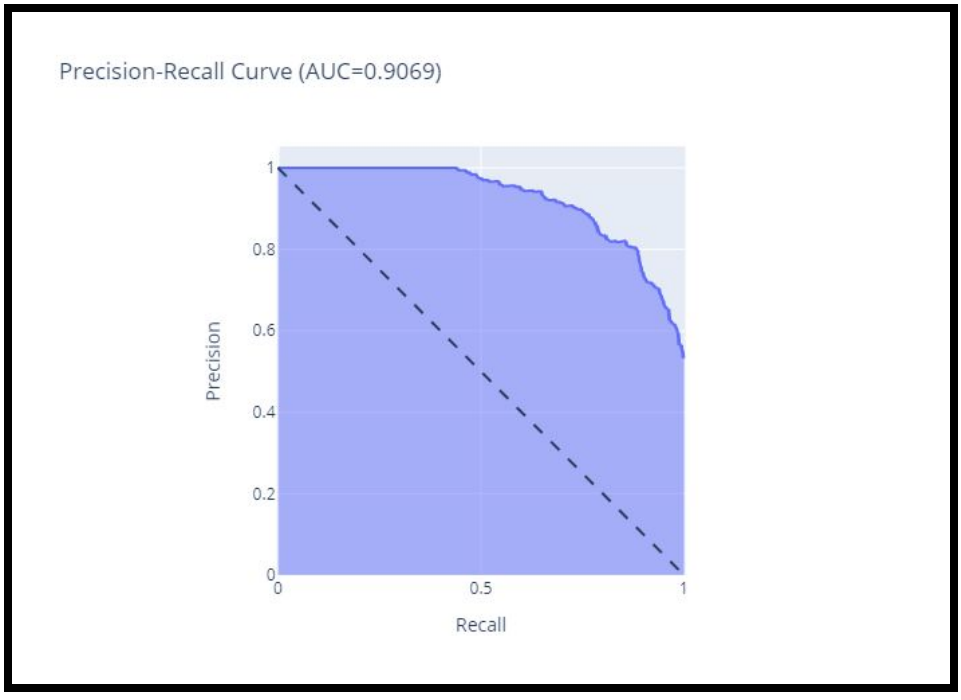
2 - For Bert, we need two things the first one is input ids and the second is attention mask which should be mapped to labels for fine tunning the Bert, so I choose max length and applied bert tokenizer and returned input ids and attention mask in tensors. (I can't choose max length to max length > 512 because for bert we can't do that).

3 - Then I simply build DataLoader and used RandomSampler for the training data and validation sampler for validation data.

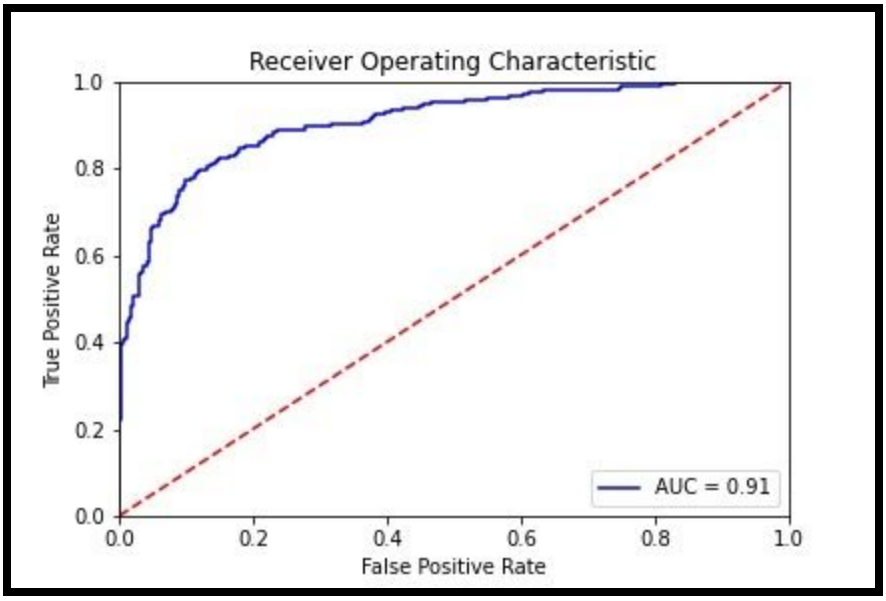
4 - After that, I build my model using bert and flatten the output from bert and used some hidden layers(768 -> 80 -> 2) and used relu activation.
5 - I also used optimizer, schedular, calculated the loss, and backpropagate, and updated the weight during the trainings.
6 - I use the batch size 32 as the author has recommended too.
7 - Then I used a trained model and calculated the various metrics for the classification such as fpr, tpr, precision, recall, F1 score.
8 - Finally use my model and calculated my prediction on the test set.

Results

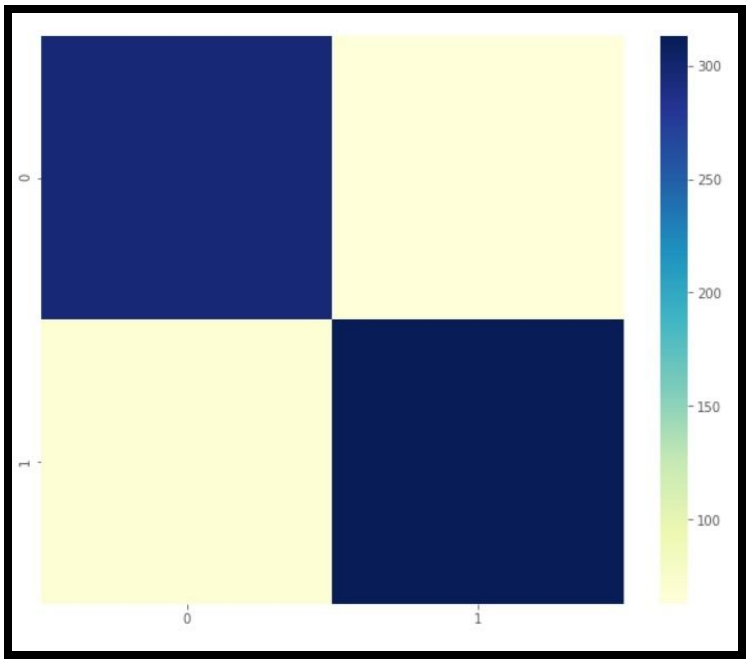
1 - Precision and Recall -



2 - ROC Curve -



3 - Confusion Matrix-



[0 0] - 297
[1 1] - 313

Score On Kaggle -

Even using very little max length and training for very few epochs the results were quite good.

Problem I faced -

1 - Some text in the train set has missing 'body' from the boilerplate which was the problem.
2 - Padding was also the issue as in the new version of Hugging face library padding is according to max length in batch rather than the overall max length
3 - Even after clearing grad I faced a lot of time "Cuda out of memory problem"
4 - Can't do it for longer text length due to which I suffer lower scores on the test dataset.

What more to do?

1 - I may have deal with the long text sequence in multiple ways -

- Used Tf-IDF and tree or linear model for classification
- Preprocess using bert tokenizer and then used RNN, LSTM for classification
- Can used TensorFlow word_tokenize, Embedding Layers, and LSTM for classification. Example

2 - Handling Cuda out of memory problems in more efficient ways.
3 - I may have trained for bigger batch sizes and more epochs.

References

1 - My Notebook from Kaggle - Most of the things are from here only.
2 - Abhishek Thakur Book - For a better understanding.
3 - For plots - For visualization
4 - Interactive Plots using Plotly - More Interactive plots and saving them
5 - This also helped me a lot - Sentiment analysis with the bert