Natural Language Processing with Disaster Tweets

Olzhas Shortanbaiuly
Nazarbayev University
olzhas.shortanbaiuly@nu.edu.kz

Abstract

Disaster prediction and classification is one of new the areas of application for the Natural Language Processing field. This work represents its solution to the Kaggle problem with disaster tweet classification using Natural Language Processing models, in the form of 5 different BERT architectures and a modified version of the best-performing architecture by the use of hypothesis tuning. Validation Loss, Validation Accuracy, Validation Recall and Validation F1-score were used as performance metrics.

1. Introduction

Natural Language Processing is a subfield of linguistics and artificial intelligence concerned with providing computers with the ability to understand text given in human language. Natural Language Processing is widely used in many applications, such as email spam detection, machine translation, information extraction, etc [7]. Other than that, natural language processing may also be used in assessing extreme weather circumstances such as heavy rainfall [5]. Therefore, the question regarding the NLP being used to predict natural disasters is raised as well, resulting in the creation of the Kaggle competition named "Natural Language Processing with Disaster Tweets".

Twitter, as a social media, is frequently used for sharing information about emergencies. Smartphone users share the emergencies observed in real life due to the ubiquity of their devices. Currently, many organizations such as news agencies or charities for helping people in need like the Red Crescent or the Red Cross are getting interested in monitoring Twitter for this reason. Despite this, not in all cases do Twitter users report something catastrophic that happens to be some real disaster. For example, the tweet "On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE" shows the sky view, even if the human eye and perception allow the human to detect that "ABLAZE" was used metaphorically but does not mean anything about disasters. Indeed, machines may be unable to detect that, as well as natural language processing models [3].

The stated Kaggle competition aims at building a natural language processing model predicting whether the tweets about disasters are truly about real disasters or not. Indeed, there were several different approaches to this competition, starting from using the classic machine learning algorithms such as logistic regression [2], followed by simple neural networks with an LSTM layer [1], and much more complex models such as BERT [4]. Different variations of BERT, such as RoBERTa are also considered as a model for this problem [6].

Among the solutions stated above, as a baseline, the BERT architecture solution (with L=12 hidden layers, a hidden size of H=768, and A=12 attention heads, 10 epochs, learning rate of 0.0001, batch size of 32) with the highest public score provided by Gunes Evitan was utilized. The project approaches to use more complex BERT architectures and vary some of the hyperparameters to achieve a higher performance.

The rest of the final report is organized as follows: the Literature Review section describes some of the papers related to the selected problem, the Dataset section describes the data itself and the data-preprocessing part of the project, while the Methodology demonstrates the baseline model, and main approach in detail with related figures and tables. The Results and Analysis section discusses the model performance and elaborates on room for improvement for the project. The Code and Submission section provides the code implementation and Kaggle submission score, and the Conclusion section finalizes the work done, with References and an Appendix also.

2. Literature Review

This topic did not frequently appear in academic papers but there are some related works discussing similar problems.

First of all, the papers in natural language processing were referred to for the purpose of looking for applications of natural language processing. Paper [7] provides a general description of concepts, methods and applications of natural language processing, while "A systematic review of natural language processing applications for hydrometeo-

rological hazards assessment" elaborates more on applying NLP to assessing extreme weather events. In this paper, the authors show how NLP can be used to leverage floods and hurricanes, and its benefits are discussed. NLP is described as a tool in data collection from social media and news articles, also, helping in fast decision-making with informed decisions taking into consideration the time constraint [5].

"Detecting Disaster Tweets using a Natural Language Processing Technique" is more relevant to this work as the author performs research on his attempt in this Kaggle competition. Suggesting his own data pre-processing algorithm with tokenization, frequency distribution, stopwords, stemming, lemmatization and POS tagging, the author proceeds with suggesting to use RoBERTa and Multinomial Naive-Bayes models as models to solve the disaster tweets classification problem. Chatbot is constructed to make predictions and the accuracy scores of 0.7946 for Multinomial Naive-Bayes and 0.7990 for RoBERTa were obtained [6]. This paper is also one of the reasons why this project focused on BERT architectures.

3. Dataset

The dataset is for a classification problem from Natural Language Processing. For the given tweet about a natural disaster, the model is expected to predict whether it is about a true disaster (class 1) or not (class 0). Data contains 10,876 hand-classified tweets: train.csv (7,613 tweets), test.csv (3,263 tweets)

Data columns:

- id (unique identifier);
- text (text/content of the tweet);
- location (location tweet was sent from, may be blank);
- keyword (particular keyword from the tweet, may be blank);
- target (real disaster or not)

3.1. Missing values

After studying both training and test sets, the presence of missing values in the *'keyword'* and *'location'* columns was observed. It was further confirmed by bar charts and the NaN values were renamed for convenient analysis as *'missing_keyword'* and *'missing_location'*.

The following observations were also made for both training and test sets, dedicating that they come from the same sample:

- *'keyword'* column has missing values of approximately **0.08%**
- *'location'* column has missing values of approximately **33.5%**

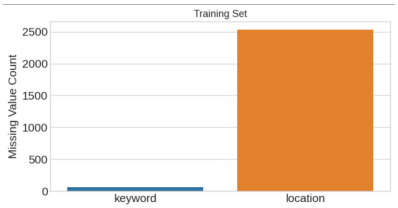

Figure 1. Missing values in training set

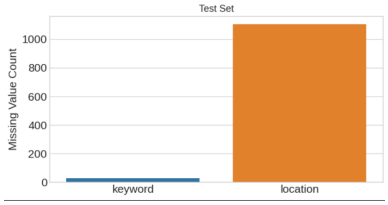

Figure 2. Missing values in test set

3.2. Distribution of values

It was determined that in *'keyword'* column, there are 222 unique values for both training and test sets, each of the training and test sets have the same set of them. It can be used as a feature based on its cleanliness seen from the output of *'value_counts'*.

Regarding the *'location'* column, the huge number of unique instances of 3342 for training and 1603 for test sets is observed. It indeed shows that they were inputs given by users. This column contains many inconsistent values such as 'MontrÌ©al, QuÌ©bec', so it can't be used as a feature.

To see the most common *'keyword'* and *'location'* values, a barplot of the most 10 common values in each of these columns was made.

As expected, *'keyword'* column most frequently contains the most common words related to natural disasters. Unfortunately, there is a considerably high number of *'missing_keyword'* values in this column.

'location' column shows some inconsistencies such as 'USA'. 'United States' and 'Los Angeles, CA' were being used as separate locations, showing some room for improvement. What is the most problematic is the majority of values in this column being *'missing_location'*, making this column even less relevant as a feature.

3.3. Target distribution

The pie chart and bar charts were made for the *'target'* column, demonstrating the distribution of the values for this column. It is depicted to be as follows:

- **0 (No Disaster)** - *57.0%* of values;
- **1 (Disaster)** - *43.0%* of values;

Balanced classes are observed so there is a low risk of bias, stratification is less critical in cross-validation and standard performance metrics are reliable.

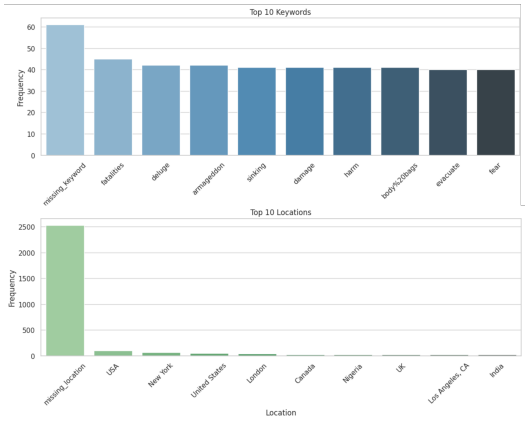

Figure 3. 10 most common values for 'keyword' and 'location' columns

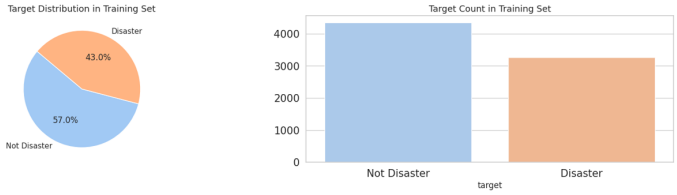

Figure 4. Target distribution

3.4. Embedding and text cleaning

3.4.1 Embeddings coverage

Train and test vocabs are created by counting the number of words in the *'text'* column for tweets, to obtain vocabulary maximally close to embeddings. In this project, the following embeddings are used for text cleaning:

- GloVe-300d-840B;
- FastText-Crawl-300d-2M;

Embedding is applied through *'build_vocab'* and *check_embeddings_coverage'* functions. *'covered'* is used to store the words that are in the intersection of embeddings and vocab. *'n_covered'* stands for total number of them. *'out_of_vocab'* is used to store the words that are in vocab but not in embeddings. *'n_oov'* stands for total number of them.

To calculate the coverage percentages, *'n_covered'* and *'n_oov'* are used. Without cleaning, both GloVe and FastTextCrawl embeddings have vocabulary coverage higher than 50% and text coverage higher than 80%. Even though both embeddings have approximately close coverage values, GloVe has higher percentages.

3.4.2 Text cleaning

Text cleaning is done through *'clean()'* function dealing with (using ReGex)

- Special characters that are attached to words removed completely;
- Contractions are expanded;
- Urls are removed;
- Character entity references are replaced with their actual symbols;
- Typos and slang are corrected, and informal abbreviations are written in their long forms;
- Some words are replaced with their acronyms and some words are grouped into one;
- Expanding informational usernames and hashtags as much as possible;

3.5. Mislabeled samples

In the dataset, 18 unique tweet instances have different labels for the duplicate tweets, probably caused by the human labeler error, also being difficult to interpret. Such tweets have *'target'* value relabeled due to its effect on accuracy.

4. Methodology

4.1. Baseline model

BERT architecture bert_en_uncased_L-12_H-768_A-12 (with L=12 hidden layers, a hidden size of H=768, and A=12 attention heads, 10 epochs, learning rate 0.0001, batch size of 32) with *'uncased'* inputs (text lower-cased before tokenization) pre-trained for English on BooksCorpus and Wikipedia used in [6] was chosen as a baseline model. It is known that the model has a Validation Recall of 0.792688, Validation F1 of 0.79343, Validation loss of 0.4521, and, most importantly, Validation Accuracy of **0.7982**.

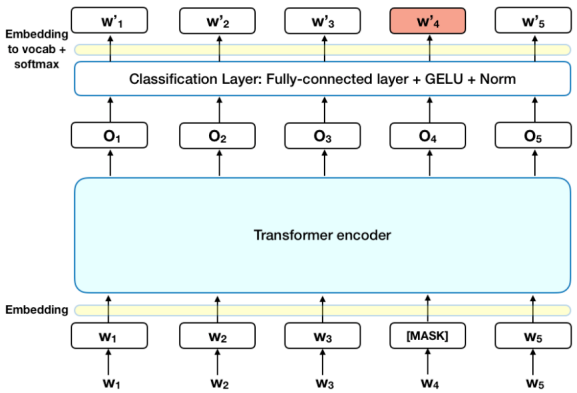

Figure 5. BERT Architecture

The baseline model is executed using 10 epochs, indeed the performance metrics converge to a constant value after 5 epochs, so for further experiments, 5 epochs were used. Other than that, a batch size of 16 also appears to be problematic, causing extreme RAM usage, therefore all implementations will be done using a batch size of 8.

4.2. Initial attempts

Different Kaggle users attempted to use simple and classic machine learning methods for classification, such as logistic regression were attempted to be used [4]. Several models were trained and **the validation accuracy** values were tracked:

- Logistic Regression (0.6845);
- SVM (0.7232);
- k-neighbors Classifier (0.62487)

These models tend to underperform in comparison to the baseline BERT model, therefore further work will be done only on BERT models.

4.3. Evaluation metric

In this Kaggle competition, **Mean F-Score** (implemented with **Macro Average F1 Score**) is the key criterion for the leaderboard. Due to the classes being balanced, there is also a need for other metrics, such as **Accuracy**, **Precision** and **Recall**, as determining the class which is more difficultly predicted is a hard task.

- **Accuracy**: proportion of correctly classified entities from the entire sample;
- **Precision**: proportion of true positive predictions from all positive predictions;
- **Recall**: proportion of true positive predictions from all the actual positive values;
- **F1-Score**: the harmonic mean of the **Precision** and **Recall**

Precision, **Recall** and **F1-Score** are usually only computed on the whole training set, it is done so as well.

ClassificationReport class includes computation of these metrics after every epoch for both the training and validation sets.

4.4. Suggested models

For this, 5 different BERT architectures from Tensorflow Hub were taken:

- **bert_en_uncased_L-24_H-1024_A-16** - text uncased before tokenization, 24 hidden layers, hidden size of 1024, 16 attention heads, without whole-word masking;
- **bert_en_wwm_uncased_L-24_H-102_A-16** - text uncased before tokenization, 24 hidden layers, hidden size of 1024, 16 attention heads, with whole-word masking;
- **bert_en_cased_L-12_H-768_A-12** - text not uncased before tokenization, 12 hidden layers, hidden size of 768, 12 attention heads, without whole-word masking;
- **bert_en_cased_L-24_H-1024_A-16** - text not uncased before tokenization, 24 hidden layers, hidden size of 1024, 16 attention heads, without whole-word masking;
- **bert_en_wwm_cased_L-24_H-1024_A-16** - text not uncased before tokenization, 24 hidden layers, hidden size of 1024, 16 attention heads, with whole-word masking;

Here **whole-word masking** means that all of the tokens corresponding to a word are masked at once during pretraining.

ClassificationReport(reporting performance metrics) and *DisasterDetector* (encoding text, building and training model, model prediction) classes were used to train the models. *FullTokenizer* class is used for input text tokenization here with *'max_seq_length'* parameter used to tune text sequence length. Model's learning process is controlled by the variables *'lr'*, *'epochs'* and *batch_size'*. After the last BERT layer, there are no other layers. As an optimizer, *'SGD'* with momentum 0.8 is used in difference to other optimizers that would not converge.

During training, plots of **Accuracy**, **Precision**, **Recall** and **F1-Score** (for validation set) versus epochs and **Training Loss/Validation Loss** loss versus epochs are made.

5. Results and Analysis

5.1. Experimental Results

Having observed these results, where **bert_en_cased_L-24_H-1024_A-16** gives the best performance metric values for **Validation Accuracy** and **Validation F1-score**, it may be aimed to further improve this model by tuning the hyperparameters. For this purpose, different values of *learning rate* and *batch_size* were tested (with *max_seq_length*, *epochs*, *activation function* hyperparameters being fixed due to specifics of the problem, e.g. *max_seq_length* not decreased to be lower than 128 considering the lengthy entries in dataset).

Among all tries, only decreasing the *learning rate* from 0.0001 to 0.00005 and increasing the *batch_size* from 8 to 16 resulted in performances that were close to the performance of the **bert_en_cased_L-24_H-1024_A-16** model. So, that means that the baseline choices of *learning_rate* and

model	Validation Loss	Validation Accuracy	Validation Recall	Validation F1-score
Baseline	0.4521	0.7982	0.792688	0.79343
bert_en_uncased_L-24_H-1024_A-16	0.3818	0.8306	0.826957	0.827910
bert_en_wwm_uncased_L-24_H-102_A-16	0.4295	0.8332	0.811493	0.820398
bert_en_cased_L-12_H-768_A-12	0.3958	0.8345	0.818175	0.825031
bert_en_cased_L-24_H-1024_A-16	0.4150	0.8398	0.832123	0.835019
bert_en_wwm_cased_L-24_H-1024_A-16	0.3989	0.8365	0.827307	0.831283

Table 1. Results with selected BERT architectures

number_of_epochs were reasonable with regard to BERT models' performance.

Therefore, **bert_en_cased_L-24_H-1024_A-16** model remains to be the best and the predictions obtained by this model will be sent to Kaggle submission.

5.2. Discussion

It was deduced that the original model **bert_en_cased_L-24_H-1024_A-16** gave the best performance indicating that there was no need to run all experiments with batch size of 16 as in the baseline solution.

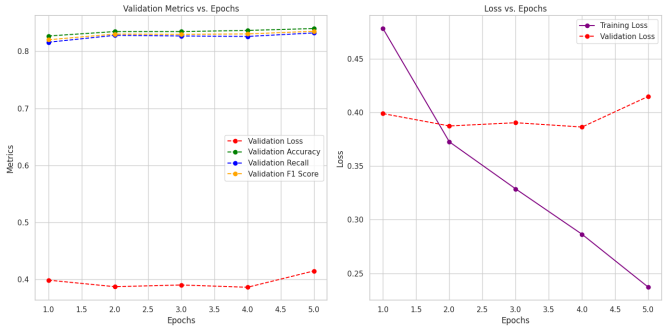

Figure 6. Performance plot of bert_en_cased_L-24_H-1024_A-16 model

Discussing the behavior of **Validation Accuracy**, **Validation Recall**, **Validation F1-scores** steadily increase as the number of epochs increases, indeed staying on the high level, starting with values higher than 0.8, indicating good performance. Other than that, **Validation Loss** remains relatively low and does not show considerable progress in value in 5 epochs, not going higher than the value 0.45, demonstrating model quality. As the number of epochs increases, **Training Loss** is decreasing dramatically, showing signs of overfitting, which can be considered in the future as room for improvement.

Due to BERT architectures being quite complex, most of the time while performing the experiments, I ran out of RAM memory, the execution was stopped and executions themselves took a long time. These models are not too convenient and can be applied only with enough computational power.

Suggested models are not efficient in terms of executed time and allocated memory (in comparison to the baseline model too), despite giving very high performance metric values. The chosen model is suggested to be used for users who prefer high accuracy to time and memory, while the baseline model, due to its comparative simplicity is suggested for users that value runtime and memory allocation more than the highest accuracy (need to still note that baseline model also gives a very high accuracy).

6. Code and Submission

The codes for this project can be found in GitHub repository.

The predicted labels obtained by the best-performing model were submitted to the Kaggle competition as *submission.csv* file, where the score of **0.78148** was obtained.

7. Conclusion

In this project, the application of Natural Language Processing to predicting disasters was studied on the example of the Kaggle competition "Natural Language Processing with Disaster Tweets". For this, the dataset from Kaggle was studied, analyzed, and preprocessed (cleaned). The embeddings of GloVe-300d-840B and astText-Crawl-300d-2M were also used to prepare the data for modeling. Then, as a solution, BERT architectures were selected with a baseline model bert_en_uncased_L-12_H-768_A-12, and 5 other BERT models were trained on our dataset, with bert_en_cased_L-24_H-1024_A-16 model giving the best performance (validation accuracy of 0.8398, validation F1-score 0.835019, all better than baseline model).

A considerably good Kaggle score of 0.78418 was obtained. Then the obtained results and performance plots were discussed along with weaknesses of the project work being determined as too complex BERT architectures, lack of computational power, and lack of efficiency in terms of time and memory. Suggestions regarding these points were made. In the future, it is planned to continue this work with different BERT architectures, such as RoBERTa, DistilBERT and AlBERT to compare their performances and even obtain a better-performing model for the task of disaster tweets classification.

model	Validation Loss	Validation Accuracy	Validation Recall	Validation F1-score
bert_en_cased_L-24_H-1024_A-16	0.4150	0.8398	0.832123	0.835019
Learning rate = 0.00005	0.4220	0.8306	0.820158	0.823520
Batch size = 16	0.4118	0.8286	0.810257	0.817095
Learning rate = 0.00005, Batch size = 16	0.3944	0.8201	0.810351	0.813563

Table 2. Results with varying hyperparameters for the best-performing BERT model

References

[1] Basic eda,cleaning and glove. https://www.kaggle.com/code/shahules/basic-eda-cleaning-and-glove#Baseline-Model. Accessed: 2023-10-24.

[2] Basic nlp on disaster tweets. https://www.kaggle.com/code/holfyuen/basic-nlp-on-disaster-tweets. Accessed: 2023-10-24.

[3] Natural language processing with disaster tweets. https://https://www.kaggle.com/competitions/nlp-getting-started/overview. Accessed: 2023-10-24.

[4] Nlp with disaster tweets - eda, cleaning and bert. https://www.kaggle.com/code/gunesevitan/nlp-with-disaster-tweets-eda-cleaning-and-bert. Accessed: 2023-10-24.

[5] M. T. Achraf Tounsi1. A systematic review of natural language processing applications for hydrometeorological hazards assessment. *Natural Hazards*, 116, 2819-1870, 2023.

[6] P. Berenice Jacqueline Sánchez Alvarado, Pedro E. Chavarrias-Solano. Detecting disaster tweets using a natural language processing technique. *ResearchGate*, 2021.

[7] K. Khatter. Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, 2022.

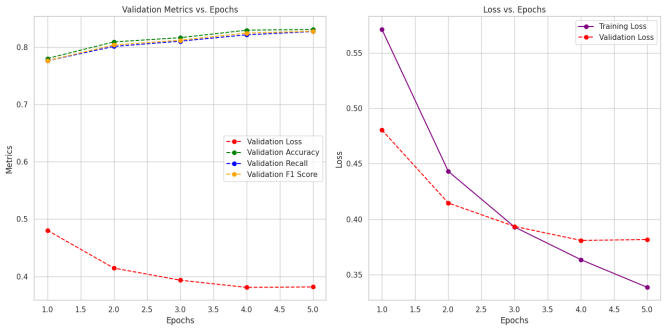

Figure 7. Performance plot of bert_en_uncased_L-24_H-1024_A-16 model

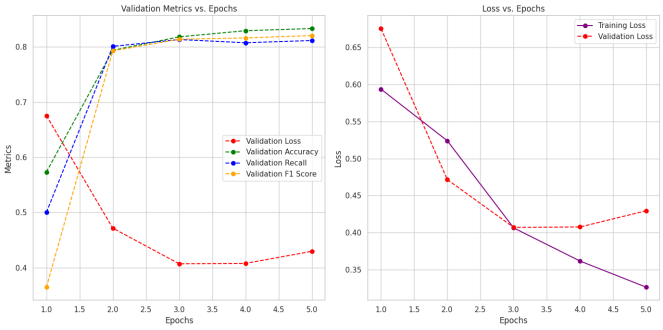

Figure 8. Performance plot of bert_en_wwm_uncased_L-24_H-102_A-16 model

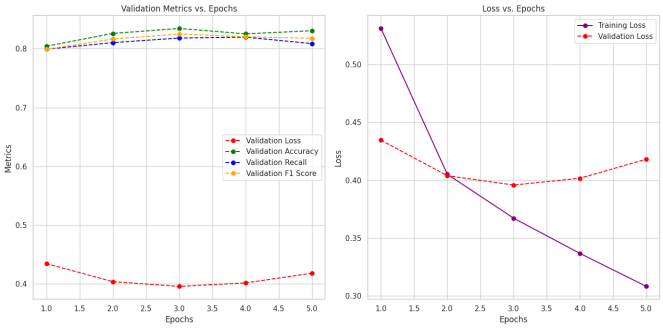

Figure 9. Performance plot of bert_en_cased_L-12_H-768_A-12 model

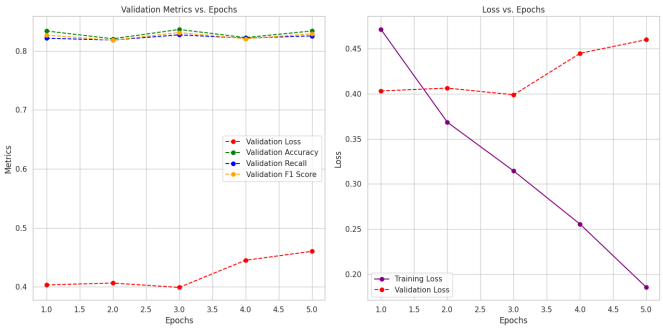

Figure 10. Performance plot of bert_en_wwm_cased_L-24_H-1024_A-16 model

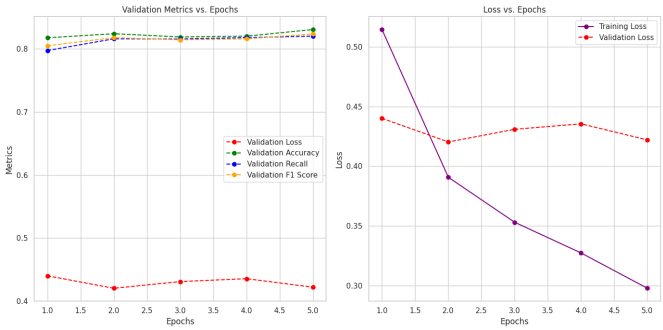

Figure 11. Performance plot of bert_en_cased_L-24_H-1024_A-16 model with learning rate of 0.00005

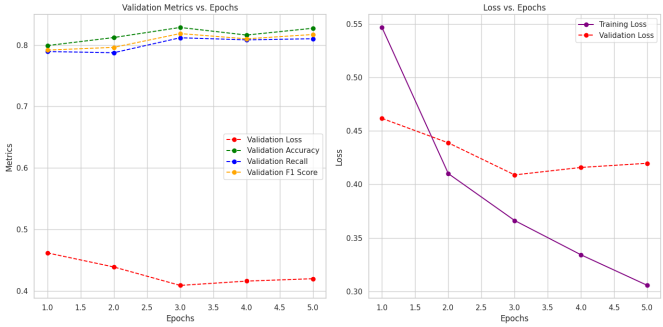

Figure 12. Performance plot of bert_en_cased_L-24_H-1024_A-16 model with batch size of 16

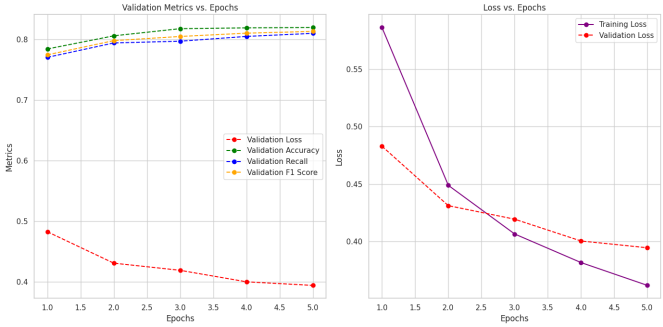

Figure 13. Performance plot of bert_en_cased_L-24_H-1024_A-16 model with learning rate of 0.00005 and batch size of 16