Exploring Graphical User Interfaces with Multiples Classifiers using BERT

Bijan Eshghi
University of Lausanne
bijan.eshghi@unil.ch

Abstract- **The proliferation of user-generated content on social media platforms and review websites has allowed for the development of robust sentiment analysis models to interpret public opinion and consumer feedback. This paper presents a comprehensive approach to the application of the Bidirectional Encoder Representations from Transformers (BERT), a state-of-the-art natural language processing model, to perform predictions on a given set of text and automate the training of different kinds of models. The methodology involves using the Gradio package to create an easy to use graphical user interface. The classification models presented are a neural network, XGBoost and the Random Forest.**

Index Terms - Natural Language Processing, Bidirectional Encoder Representations from Transforms, Twitter, IMDB, Reddit, Yelp, XGBoost, Neural Network, Random Forest.

I. Introduction

Natural Language Processing (NLP) has seen remarkable advancements in recent years, significantly driven by the development of sophisticated machine learning models. Among these, the Bidirectional Encoder Representations from Transforms (BERT) model, introduced by Google in 2018, represents a substantial leap forward (Chang, Lee, Toutanova 2019). BERT's innovative architecture, grounded in the Transformer model, enables it to understand the context of words in a sentence by examining them in both directions- forward and backward- simultaneously. This bidirectional approach contrasts sharply with previous models, which typically processed text in a unidirectional manner.

BERT's introduction has revolutionized various NLP tasks, including question answering, sentiment analysis, and named entity recognition (to name a few), by providing a pre-trained model that can be fine-tuned for specific tasks with minimal data and computational resources. Its ability to capture nuances of human language, such as polysemy and syntactic ambiguity, has set new benchmarks in NLP performance (Kacprzak 2024). This paper is interested in the application of BERT with multiple classification methods in the context of a graphical user interface (GUI) for both training and prediction.

II. Research Question

As such, this paper is interested in the following research question: *How can a graphical user interface be designed to effectively facilitate the training of BERT models with various classifiers and the execution of predictions?*
It is important to explore such a question as designing an effective GUI makes the powerful capabilities of BERT modes accessible to a broader audience, including those who may not have extensive programming skills. A well-designed GUI can also streamline the workflow for training BERT models. Furthermore, it can help minimize errors by providing structured and guided processes for setting parameters, preprocessing data, and interpreting results. Finally, it is possible that such a GUI can serve as a valuable educational tool, helping users understand the underlying processes of training and using BERT models.

III. Methodology

In order to answer this research question, three Python packages for creating a GUI were considered. The first was Tkinter, a very popular package for creating general-purpose GUIs in Python. Being easy to use and highly customizable, it can most certainly be used to great effect for creating a final complete user interface for machine learning applications, such as what is being considered in this paper. However, as useful as customizability is for the purpose of developing GUIs, in this case it would come as a disadvantage since much time would have to be spent on creating a visually appealing and easily usable interface as an addition to working on efficient functionality, which is the focus of this paper. Furthermore, we are more interested in creating a well functioning demo for this project, not necessarily a finished complete product. The second package that was considered was Streamlit, a web-based application that is designed for users wanting to create machine learning GUIs. This package was preferred over Tkinter due to its specialization for machine learning, however having to depend on a web browser for development implied other complications. Having to depend on an internet connection, possible resource limitations, security concerns as well as an overall familiarity with a new web-application were ultimately the reason why this package was dropped as an option for development. As such, the Gradio package was chosen for development.

Gradio is a Python library that simplifies the creation of user interfaces for machine learning models. While not as popular as the other two packages, Gradio allows developers to quickly build interactive web interfaces with minimal code, allowing users to input data, interact with models, and visualize outputs in real time. One can also choose to run the application locally. Gradio supports a wide range of input and output types, such as text, images, audio, and video, making it ideal for machine learning tasks.

IV. Code implementation

A. *Task 1: Sentiment Prediction*

The first interface enables users to predict sentiments using pre-trained models for three different tasks: Emotion Identification, Positive/Neutral/Negative Sentiment and Star Rating prediction. Users can choose between two data sources: a pre-allocated dataset or their own dataset. The pre-allocated datasets include Twitter data for emotion prediction, Reddit data for positive/neutral/negative classification, IMDB data for positive/negative classification, and Yelp data for star ratings. For more information on these datasets and how they were trained, please refer to the other paper for the course Advanced Data Analysis. Users also select the classification method they wish to use and the trained model to be used for prediction. The classification methods are Neural Net, Random Forest, or XGBoost.

Depending on the selected classifier, the appropriate model is loaded from a pickle file. The neural network predictor, random forest predictor, or XGBoost predictor functions are then invoked to process the input text and provide sentiment predictions. If users opt to use their own dataset, the interface loads the first model available in the "Your_Pickle" directory for the chosen classification method and performs sentiment prediction using this model.

B. *Task 2: Train Your Own Model*

The second interface allows users to train a sentiment analysis model on their own data that they can upload via the "Upload a File" button. Users select a classification method and upload a data file (in .csv, .tsv, or .parquet format) containing the text and label columns. The uploaded file is processed using the corresponding BERT-based model training functions.

The interface guides users through the training process, ensuring that the uploaded dataset meets the required format and providing feedback on the training progress (the latter only with access to a terminal). As previously mentioned, the trained model is saved in the "Your_Pickle" folder upon completion, making it available for future predictions in Task 1.

V. Maintaining and updating codebase

A. *Task 1*

The last point mentioned for task 1 above is the first main flaw of this task, as a user may wish to train multiple dataset with the same classifier and compare predictions. The second flaw, while not quite as limiting, pertains to the fact that the user can currently choose to, for example, predict emotions on a model that was trained to predict star rating. While this can be avoided by simply choosing the correct trained model, the application would ideally be able to somehow "gray out" or disable the usage of an incorrect model.

Finally in order to run predictions, the original training scripts from the ADA project must be included in the environment. This is because the pickle files were trained on the original .py files (BERT_NN.py, for example). Unfortunately, upon realizing that in order to fix this issue, I needed to train all the models on the new .py files (BERT_NN_GUI, for example), I had run out of time to train new models. While this issue is a significant problem, it can easily be fixed with time by training new models and saving their pickle files using the new training scripts. It is important to resolve these three issues, however the complications presented for task 2 are more urgent.

B. *Task 2*

There are a few major unresolved issues for task 2. The most important one, as mentioned in section IV, is the fact that the interface only provides feedback if a terminal is available, or alternatively, if it is done in a notebook. This has been an issue since the early stages of development for this GUI. Changing the code in BERT_NN.py, for example, to make the program return a single confusion matrix does not resolve the issue, despite there being only one image as an output (this can be done by setting evaluate = FALSE and test=TRUE for the training function). Furthermore, due to the limited size of the community that uses Gradio, there are not many online discussions/threads that are able to assist in fixing this issue. As such, users wishing to see feedback on the training of their model without access to a terminal or using a notebook will be unable to do so for the time being. This is, by a significant margin, the most important and first issue that should be resolved in future updates to the codebase. The second complication for task 2 is the lack of customizability of the classification methods. At the moment, the classification methods are pre-set, and if a user wishes to tweak parameters they are forced to go into the program files and change them manually. Depending on the user base this application would attract, this problem is likely the next issue that should be resolved in future updates to the codebase.

VI. Results

Despite the complications presented in section V, the graphical user interface developed in this paper can complete tasks such as sentiment prediction with multiple classifiers and data types as well providing the user the ability to train their own model and run predictions on it (see pictures 1 and 2 in the appendix for an example of each task).

VII. Conclusion

The development of a GUI for training and using BERT models with various classifiers has the potential to democratize access to advanced natural language processing (NLP) capabilities. By using Gradio, an interface that simplifies the complex processes of training and prediction was created, making these functionalities accessible to users with limited programming skills. The application however, contains limitations, such as dependency on a terminal for training feedback and limited customizability of classification methods. Moreover, addressing the customization limitations and ensuring seamless integration of training scripts will significantly enhance user experience and functionality.

In summary, this project lays a solid foundation for an accessible and efficient GUI for BERT model training and prediction. Future work will focus on refining the interface to resolve existing issues and incorporate additional features, ultimately making advanced NLP accessible to a wider audience

VIII. References

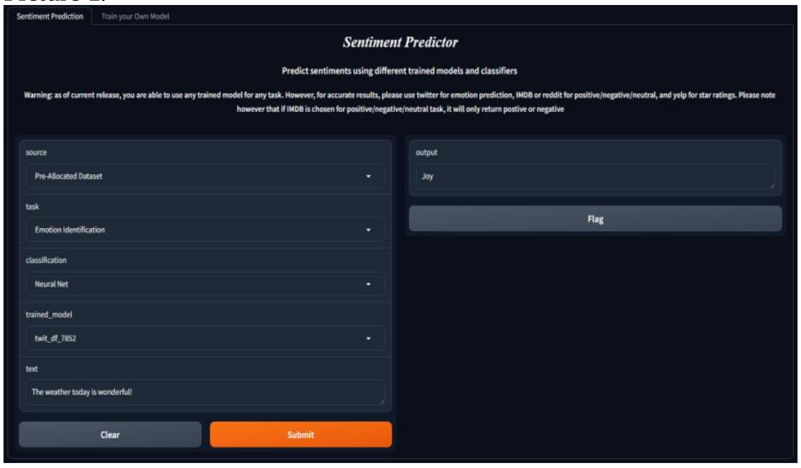
Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May 24). Bert: Pre-training of deep bidirectional Transformers for language understanding. arXiv.org. https://arxiv.org/abs/1810.04805

Kacprzak, K. (2024, January 26). Roberta vs. Bert: Exploring the evolution of Transformer models. Data Engineering, MlOps and Databricks services. https://dsstream.com/roberta-vs-bert-exploring-the-evolution-of-transformer-models/#:~:text=BERT’s%20pre%2Dtraining%20involves%20two,language%20inference%2C%20and%20sentiment%20analysis.

IX. Appendix

ChatGPT was used to assist in writing this paper in a few sections. It was not, however, used to program anything anywhere in this project.

Picture 1:



Sentiment Predictor

Predict sentiments using different trained models and classifiers

Warning: as of current release, you are able to use any trained model for any task. However, for accurate results, please use twitter for emotion prediction, IMDB or reddit for positive/negative/neutral, and yelp for star ratings. Please note however that if IMDB is chosen for positive/negative/neutral task, it will only return postive or negative

source: Pre-Allocated Dataset

task: Emotion Identification

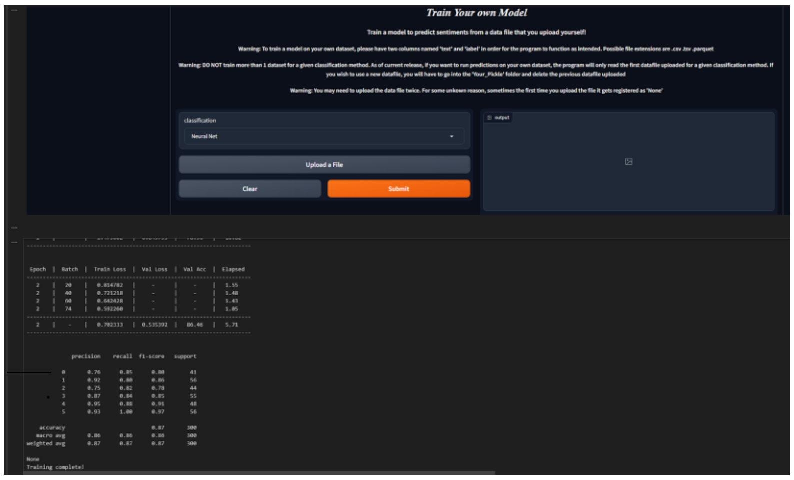
classification: Neural Net

trained_model: twit_df_7852

text: The weather today is wonderful!

output: Joy

Picture 2:



Train Your own Model

Train a model to predict sentiments from a data file that you upload yourself!

Warning: To train a model on your own dataset, please have two columns named 'text' and 'label' in order for the program to function as intended. Possible file extensions are .csv .tsv .parquet

Warning: DO NOT train more than 1 dataset for a given classification method. As of current release, if you want to run predictions on your own dataset, the program will only read the first datafile uploaded for a given classification method. If you wish to use a new datafile, you will have to go into the 'Your_Pickle' folder and delete the previous datafile uploaded

Warning: You may need to upload the data file twice. For some unkown reason, sometimes the first time you upload the file it gets registered as 'None'

classification: Neural Net

```
 Epoch  |  Batch  |  Train Loss  |  Val Loss  |  Val Acc  |  Elapsed
----------------------------------------------------------------------
   2    |   20    |   0.814782   |     -      |     -     |   1.55
   2    |   40    |   0.721218   |     -      |     -     |   1.48
   2    |   60    |   0.642428   |     -      |     -     |   1.43
   2    |   74    |   0.592260   |     -      |     -     |   1.05
----------------------------------------------------------------------
   2    |    -    |   0.702333   |  0.535392  |   86.46   |   5.71
----------------------------------------------------------------------


              precision    recall  f1-score   support

           0       0.76      0.85      0.80        41
           1       0.92      0.80      0.86        56
           2       0.75      0.82      0.78        44
           3       0.87      0.84      0.85        55
           4       0.95      0.88      0.91        48
           5       0.93      1.00      0.97        56

    accuracy                           0.87       300
   macro avg       0.86      0.86      0.86       300
weighted avg       0.87      0.87      0.87       300

None
Training complete!
```