Transformers in NLP using RoBERTa and XLNet

Business Objective

In Part 1 of the Transformer series lectures - Multi-Class Text Classification with Deep Learning using BERT), we have done an in-depth analysis of NLP.

We started from simpler NLP models like Bag of Words (BOW), TF-IDF and moved towards word embedding models like Word2Vec, Glove, and then to simple/bi-directional RNNs. Finally, we came across complex Transformer architecture and witnessed the power of BERT, one of the highest performing State-of-the-Art Transformer models.

In Part 2 of this series, we will discuss two more novel architectures which worked on improving over BERT performances using different training & optimization techniques.

These are:

- RoBERTa: A Robustly Optimized BERT Pretraining Approach
- XLNet: Generalized Autoregressive Pretraining for Language Understanding

We will analyse the architectures of these two models, study their training and optimization techniques and finally use them to classify Human Emotions into separate categories.

Data Description

Emotion is a dataset of English Twitter messages with six basic emotions: anger, fear, joy, love, sadness, and surprise. This dataset is taken from the hugging face library. (https://huggingface.co/datasets/emotion)

The dataset comprises of three data categories,

- Train - 16000 rows and 2 columns
- Validation - 2000 rows and 2 columns
- Test - 2000 rows and 2 columns

The two columns are labels and text.

- 0: sadness
- 1: joy
- 2: love
- 3: anger
- 4: fear
- 5: surprise

Aim

The project aims at building two models, namely RoBERTa and XLNet to perform classification on the human emotion dataset.

Tech stack

- ➢ Language - Python
- ➢ Libraries - datasets, numpy, pandas, matplotlib, seaborn, ktrain, transformers, tensorflow, sklearn

Environment

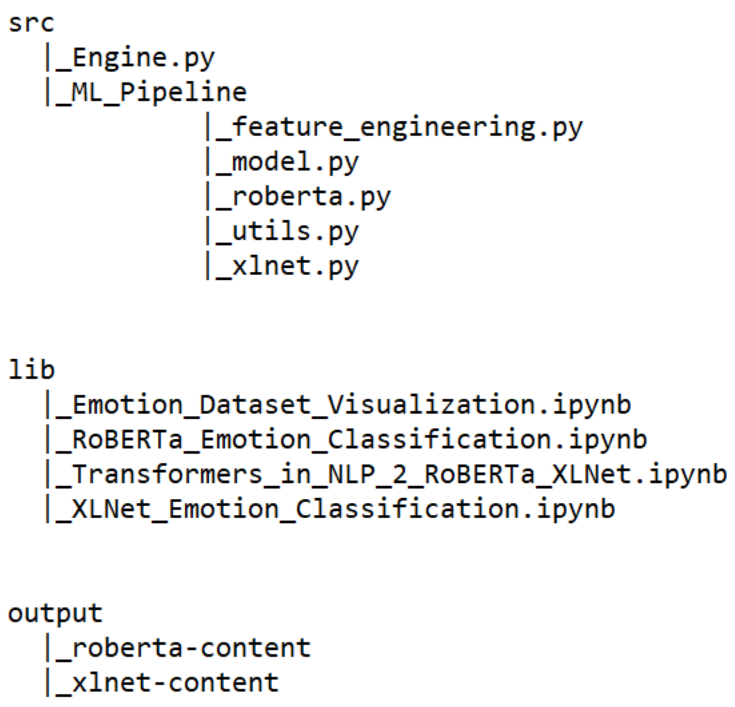
- ➢ Jupyter Notebook
- ➢ Google Colab Pro (Recommended)

Approach

1. Install the required libraries
2. Load the ‘emotion’ dataset
3. Read the dataset across all the three categories
4. Convert dataset object to data-frame and create a new feature
5. Data Visualization
    - Histogram plots
6. RoBERTa model
    - Create a RoBERTa model instance.
    - Split the train and validation data
    - Perform Data Pre-processing
    - Compile RoBERTa in a K-train learner object
    - Find optimal learning rate
    - Fine-tune the RoBERTa model on the dataset
    - Check for performance metrics
    - Save the RoBERTa model
    - Use the RoBERTa model on the test data and check for the performance.
7. Understand the Autoregressive and Autoencoder models
8. XLNet model
    - Load the required libraries
    - Create an XLNet model instance
    - Split the train and validation data

- Perform Data Pre-processing
- Compile XLNet in a K-train learner object
- Find optimal learning rate
- Fine-tune the XLNet model on the dataset
- Check for performance metrics
- Save the XLNet model
- Use the XLNet model on the test data and check for the performance

Modular code overview



```
src
  |_Engine.py
  |_ML_Pipeline
            |_feature_engineering.py
            |_model.py
            |_roberta.py
            |_utils.py
            |_xlnet.py


lib
  |_Emotion_Dataset_Visualization.ipynb
  |_RoBERTa_Emotion_Classification.ipynb
  |_Transformers_in_NLP_2_RoBERTa_XLNet.ipynb
  |_XLNet_Emotion_Classification.ipynb


output
  |_roberta-content
  |_xlnet-content
```

Once you unzip the modular_code.zip file you can find the following folders within it.

1. src

2. output

3. lib

1. Src folder - This is the most important folder of the project. This folder contains all the modularized code for all the above steps in a modularized manner. This folder consists of:
    - Engine.py

- ML_Pipeline
  The ML_Pipeline is a folder that contains all the functions put into different python files which are appropriately named. These python functions are then called inside the Engine.py file.

2. Output folder - The output folder contains the best fitted model that we trained for this data. This model can be easily loaded and used for future use and the user need not have to train all the models from the beginning.
   Note: This model is built over a chunk of data. One can obtain the model for the entire data by running Engine.py by taking the entire data to train the models.

3. Lib folder - This folder contains the original Ipython notebooks that we saw in the videos. There is another reference folder present, which contains all the examples explained during the videos.

Project Takeaways

1. Understanding the business problem.
2. Transformer Architecture
3. Concept of Self Attention
4. BERT model overview
5. Understanding of the RoBERTa model (A Robustly Optimized BERT Pretraining Approach)
6. Various approaches for the RoBERTa model
7. RoBERTa model Architecture
8. Import the data from the hugging face library.
9. Perform data pre-processing on the data.
10. Build a RoBERTa model instance
11. Compile and fine-tune the RoBERTa model
12. Autoregressive and Autoencoder models
13. Understanding of the XLNet model
14. XLNet architecture vs. BERT architecture
15. Compare XLNet with BERT and RoBERTa
16. Build a XLNet model instance
17. Compile and fine-tune the XLNet model
18. Evaluate the models based on performance metrics
19. Evaluate the models on unseen data (test data)
20. Save the models