AG News classification using BERT model

Business Objective

The most abundant data in the world today is in the form of texts. Text is a rich source of information, but extracting insights from it can be complex and time-consuming as they are in an unstructured format. Hence, having a powerful text-processing system is critical and is more than just a necessity. There are various algorithms designed for performing the text classification today, BERT being one of the most popular.

Bidirectional Encoder Representations from Transformers or BERT is a prevalent NLP model from Google known for producing state-of-the-art results in a wide variety of NLP tasks. The importance of Natural Language Processing (NLP) is profound in the artificial intelligence domain.

The BERT algorithm is built on top of breakthrough techniques such as seq2seq models and transformers. The seq2seq model is a network that converts a given sequence of words into a different sequence and is capable of relating the words that seem more important.

This project will cover in detail the application of the BERT base model concerning text classification. We will witness how this state-of-the-art Transformer model can achieve extremely high-performance metrics for a large corpus of data comprising more than 100k+ labelled training examples.

Data Description

For our case study, we will be using the datasets from the hugging face library.
The BERT model will be built on the AG News dataset.

- AG News (AG’s News Corpus) is a sub dataset of AG's corpus of news articles constructed by assembling titles and description fields of articles from the 4 largest classes
- The four classes are: World, Sports, Business, Sci/Tech
- The AG News contains 30,000 training and 1,900 test samples per class.

Aim

The project aims at building, training and fine tuning the BERT model with respect to classification on the AG News dataset.

Tech stack

- Language - Python
- Libraries – ktrain, transformers, datasets, numpy, pandas, tensorflow, timeit

Environment

- Jupyter Notebook

Approach

1. Checking the hardware acceleration settings.
2. Installing the required libraries
3. Checking for the available dataset from the hugging face library
4. Importing the required dataset
5. Loading the train and test data
6. Creating dataframe objects for train and test data.
7. Performing data pre-processing
8. Creating the BERT model.
9. Compile the BERT model.
10. Training the BERT model on some defined hyperparameters.
11. Evaluating the performance metrics
12. Saving the model.

Modular code overview

```
src
  |_engine.py
  |_ML_Pipeline
            |_feature_engineering.py
            |_model.py
            |_utils.py


lib
  |_Bert_AG_News_Classification.ipynb
  |_Transformers_in_NLP_1_Explained.ipynb


output
  |_tf_model.preproc
```

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 is a reference folder. It contains the original Ipython notebook that we saw in the videos.

Project Takeaways

1. Understanding the business problem.
2. Understanding basics of NLP
3. Understanding the Bag of Words (BOW) and TF-IDF models
4. Understanding word embedding techniques like Word2Vec, Glove and FastText models.
5. Understanding the RNN and LSTM models.
6. Understanding the concept behind transformers, encoder-decoder architecture.
7. Understanding the Attention mechanism.
8. Understanding the BERT model.
9. Learning the special tokens used in the BERT model.
10. Understanding the BERT pre-training on two NLP tasks, Masked Language Modelling (MLM) and Next Sentence Prediction (NSP)
11. Importing required libraries.
12. Importing the data from the hugging face library.
13. Creating dataframe objects for train and test data
14. Performing data pre-processing on the dataframes.
15. Creating the BERT model.
16. Compiling and training the BERT model in a learner object.
17. Evaluating the performance metrics
18. Learn how to save the model.