Sentence Inference Challenge

About the Challenge:

For every given pair of sentences -- (sentence-1, sentence-2), you need to determine if sentence-2 can be logically inferred given sentence-1.

Data Set Available:

The corpus is a collection of human-written English sentence pairs manually labelled for balanced classification with the label's entailment, contradiction, and neutral. There are 9350 training and 494 testing sentence pairs.

Notebook:

Shows EDA of the data and how the model was trained from importing the data to saving the model. Also contains the part where you can reload the model and test file to get the predicted data as an output.

Libraries used:

- Tensorflow
- os
- io
- Pandas
- numpy
- sklearntransformers
- official
- seaborn
- tensorflow_hub

Findings of EDA:

1. Number of unique sentences (Sentence1 and sentence2 individually considered) = 12501
2. Number of duplicate sentences (Sentence1 and Sentence2 considered in pair) = 1
3. Percentage in each class:
    - Neutral – 32.83

- Entailment – 33.86
- Contradiction – 33.31

Assumptions:

1. Duplicate Sentences (Sentence1 an Sentence2 considered in pair) as assumed to be 0.
2. Classes are approximated balanced.

Approach:

First the training was tried on the Glove encodings and LSTM model. Stop words were removed, for normalisation purpose stemming ang lemmatization was done, then the glove encoding of the sentences were fed in 2 separate model having tanh activation function with final layer having sigmoid, and then output of these 2 models were merged into other model having relu activation and last layer sigmoid. Maximum accuracy was approx. 48%. Same was tried with Fasttext encodings but not substantial increase of accuracy was observed.

As semantic, syntactic, and contextual understanding was required, so rather than training model from scratch, we decided to use pre trained Bert model for tokenization, encoding and classifying because as BERT is language model, which is bidirectionally trained, has 24 layers (transformer blocks), 16 attention heads and 340 million parameters, which can have a deeper sense of language context and flow than a model being trained from scratch.

Data Pre-processing:

Dependent Variable:

- As the data is categorical with 3 classes, we convert it into one hot encoded labels cause in case of ordinal labels model can get confused that the categories are having some order in it and as our labels are not ordinal, so we one hot encode them.

Independent Variable:

- Stemming and Lemmatizing are not done because it changes the grammatical structure of the words (e.g., verb form)
- Vectorizer like Word2Vec were not used because they do word level encoding and here whole sentence level was needed and do2Vec or Fasttext could have

been used, but Bert encoding gives the best sentence encoding with contextual and sentence flow understanding.

- Data was split into train, test, and validation data by doing stratified sampling so that there is no unbalancing induced in data.
- Bert tokenizer was initialized and then sentence encoding was done to the pair of sentences by arranging them in the format “[CLS] sentence1 [SEP] sentence2 [SEP]” to get **token id’s** (They *are token indices, numerical representations of tokens building the sequences)*, **input id’s** (binary mask identifying the two types of sequence in the mode) and **attention masks** (separate sentence sequence by paddings) which are then given as an input to our model.

Model:

1. Configuration of pretrained model is downloaded.
2. Epochs and Bath size are selected.
3. Optimiser used is AdamW with learning rate 2e-5
4. Loss, call-back and metrics are defined, and model is compiled and trained using encoded training data.
5. SoftMax function is defined to apply on the output from the model and then it is converted to one hot encoded form.
6. Training is performed.

Model Parameters Used –

<u>Optimiser</u>: ADAMW as **AdamWeightDecay** is the **best** among the adaptive **optimizers** in most of the cases. Big steps when the gradients do not change much and small steps when they vary rapidly to reach minima of cost function.

<u>Learning Rate</u>: .00002 (tried 0.0001, 0.00005, 0.00001)

<u>Batch Size</u>: 32 (tried 20, 40) as higher batch sizes lead to lower asymptotic test accuracy that’s why we started from low BS.

<u>Callbacks:</u> Early stop callback is given because, if training loss keep on decreasing and validation loss starts to increase, model may get overtrained which may lead to more variance.

<u>Loss</u>: Categorical Cross entropy is used because it is multi class classification. Binary cross entropy would have been used if it was multi label or binary class type classification.

<u>Metrics</u>: Categorical Accuracy as the classification is categorical

Activation SoftMax: converts the scores to a normalized probability distribution which helps us to identify the most probable class of the record.

Achieved Accuracy-

Validation Accuracy – 79.88%

Training Accuracy – 92.62%

After Model Training

1. After training model, the model is tested using the encoded test data by using F1 Score.
2. Model is saved by name 'saved_model'.
3. Rest is made such that it can handle new data for prediction, so model is recalled, and data is predicted for the unknown test sample of 493 records.
4. Predictions are saved as csv file 'predicted.csv'

NOTE: Operations like 'One Hot Encoding, Inverse One Hot Encoding, and Softmax are done by custom models declared with the libraries. The dictionary for label conversion 'tamp' is also declared with the libraries. Please kindly do not change it.

Reference:

https://www.tensorflow.org/official_models/fine_tuning_bert

https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff

https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd

https://www.tensorflow.org/official_models/fine_tuning_bert

https://www.kaggle.com/vsmolyakov/keras-cnn-with-fasttext-embeddings

https://medium.com/atheros/text-classification-with-transformers-in-tensorflow-2-bert-2f4f16eff5ad

https://www.kaggle.com/dhruv1234/huggingface-tfbertmodel

https://keras.io/api/callbacks/early_stopping/

www.medium.com

www.analyticsvidhya.com