Classifying the emotions using Bert Model.

We know that this course did not cover any deep learning models on NLP but we want to experiment on the capabilities of deep learning and Large Language Models (LLMs) on classifying the text. In this experiment we tried to classify the Go Emotions dataset into 3 different categories “Positive”, “Negative”, “Neutral”.

Using transformers module we have used a simple **BertForSequenceClassification** prebuild model and trained and evaluated the model with the Go Emotions train and test datasets respectively. We followed the below procedure create, train, and evaluate the model.

1. **Importing the datasets:** In the initial steps we imported the train and test data from CSV files into python using pandas. And we sliced the imported train data into train and validation data with 90% data to train and 10% data to do cross validation during the training phase.
2. **Word Embeddings:** In the machine learning models the machine will not understand any words, it will understand only numbers. So, in order to change the word to list of numbers (which are called as word embeddings) we used **BertTokenizer** from transformers module to convert our processed text to word embeddings.
3. **Converting to Tensor Dataset:** To train the Bert model in pytorch we need to send the data in a required format to do that we used **TensorDataset** method to convert the word embeddings to a tensorDataset.
4. **Defining the Model:** In this experiment we just used the prebuilt Bert Model (**BertForSequenceClassification**) that was provided by the **Transformers** Module. Here we want to do transfer learning by using the weights in the prebuilt model and update the weights based on the go emotions data.
5. **Train the Model:** To train the model I passed the above created TensorDatasets and predicted the output of the model and based on the loss function and the predicted outputs the weight is updated in order to minimize the loss.
6. **Validate Using Test Data:** Once we have trained the model, we used the Go emotions test dataset to evaluate our model.

Note: We initially tried to run the above model in my **local machine**, but it showed around **9 hours to complete 1 epoch**. So I used the Google Colab to utilize its free GPUs. In colab after using GPU the training time has decreased to just around 45 minutes for 1 epoch. Due to Colab limitation on the GPU access time **we just trained our model for only 3 epochs.**

Some stats on our training data:

1. The training data is highly unbalanced we have 53% positive texts, 30% negative texts, 16% Neutral texts.
2. After training we got the below accuracies on the validation data   
   Class: positive - Accuracy:1509/1741 = 0.8667432510051695

Class: negative - Accuracy:726/984 = 0.7378048780487805

Class: neutral - Accuracy:339/537 = 0.6312849162011173

Observations on Test Data: (The test data is not included in my training process)

1. The model performed decently on all the three labels i.e.. on (Positive, Negative, Neutral) with accuracy of 90.00% (1895/2104) on Positive texts, 76.54% (966/1262) on the Negative texts, 64.99(440/677) on Neutral Texts
2. As the data is highly imbalanced overall accuracy does not show the true performance of the model, so i calculated the F1 score. The F1 Score for the model that ran on test data is around 0.8117. For this f1 score the baseline model is performing decently in classifying the texts
3. The Confusion matrix for the test data predictions  
   A graph with green squares and numbers

   Description automatically generated

Note to Zoe/ Ademola: while adding this image to org file the image reference is present in

./images/test\_confusion\_matrix.png

1. The model is slightly biased towards the Positive comments. From the confusion matrix we can see that most of the mis-classified Negative and Neutral comments are classified as positive comments. (It is expected as we have a greater number of positive data and due to GPU restraint, I trained only for 3 epochs. Based on the losses from each epoch the model still not converged after 3 epochs which shows the capability of the model to perform well on further training).

(Note to professor: We know that you will not consider anything that is not part of this org file. We were not able to integrate our deep learning changes with the sentiment class as we were struck on the issue where we need to save the model state on the google colab and load it in local machine to correctly predict the text. But while loading the model state in my local we were getting some issue with the model state. Due to time constraint, we were unable to fix the issue in time. But if you want to explore and play around with the Bert implementation, please feel free to use the IPYNB file present in **./experiments/Vamsi\_Gautham\_exp\_emotion\_detection.ipynb** file. The file consists of detailed explanation, implementation, and results of our Bert experiment).

Future Scope:

1. To increase the performance of the model the model needs to be trained further by increasing the number of epochs.
2. The model can be fine-tuned by changing the learning rate (lr), number of trainings steps.
3. To further classify the emotions, we can create a two or multistage ensembled model which further classify the text into minute detail emotions on the second or further stages (based on the ensemble)