CE 314/887 Assignment 2

Text classification December 2022 PRID – RAVIN39808

**Introduction :**

           This project aims to do sentiment analysis on IMDb movies data to train a machine learning model which will be able classify positive and negative reviews. Trained on data from imdb.com by making a generalized model perform effectively on unseen data rather than the one which would memorize from data. There is a vast number of models which could achieve this task, In this project, However, We will take the Deep learning Rnn approach which we will discuss in detail in this report.

**Project Dependencies :**

           Before importing data and analysing it we need to import necessary packages/modules/libs/. Since we have taken a Deep learning approach we will import and use PyTorch which possesses a huge number of easily accessible APIs to build state-of-the-art machine learning and deep learning models. Moving to data manipulation we use NumPy and pandas for those purposes. For visualization, We use seaborn and matplotlib. Those are the major imports which are rudimentary to any modern machine learning project life cycle. After completing the above steps, We load the IMDb dataset with the pandas read\_csv command, which converts CSV data into pandas data frame.

**Data Pre-processing and Exploratory Data Analysis:**

           In this part of our project, We dig deep into data cleaning, shaping and Normalizing. Firstly we remove duplicate data and handle missing values and then we split the data into training and test set of 80% and 20% to avoid data leakage. After this, We remove HTML tags, non-alpha chars, whitespace most specifically stop words. After this process, We convert every token in our data to lowercase and apply tokenization. After going through all this we will get clean data for visualization purposes. To see linguistic features and relationships between different entities on the dataset. We start data visualization by looking most frequent words used in both positive and negative reviews using word Cloud by creating a cloud of words. Then we look at, Number of chars and words in the text and the average number of words in a review to better understand our data. To gather a combination of words that occurs most frequently together we perform n-gram analysis from uni, bi and Tri-gram. Finally, we conclude by visualizing the length of each review in our data. After the visualization process, We pad every sequence to the maximum length as 500

Since we have a low number of reviews length going beyond 500 thus we conclude with 500 as our maximum sequence length. Observations : Mean review length = around 69.minimum length of reviews is 2. There are quite a few reviews that are extremely long, we can manually investigate them to check whether we need to include or exclude them from our analysis.

**Building the Pipeline :**

           In this part, We convert the data from the NumPy array to the TensorFlow data set object. With a batch size of 50, we load our data to train and validation variables using the function Data Loader. We need to add an embedding layer because there are fewer words in our vocabulary. It is massively inefficient to one-hot encode that many classes. So, instead of one-hot encoding, we can have an embedding layer and use that layer as a lookup table. You could train an embedding layer using Word2Vec, then load it here. But, it's fine to just make a new layer, using it for only dimensionality reduction, and let the network learn the weights. Building the architecture for the Rnn Lstm model named sentiment Rnn in which we specify output and hidden dimensions and add embedding layer to it. With those, We also add a dropout layer that randomly inactivates some of the neurons in the network to avoid overfitting. Thereafter, We start building a forward function for our list with dropout and a fully connected layer. In this model we use the sigmoid function at the end we return the last sigmoid output which is the activation value of the last layer’s neuron ( output layer ). Moving on to hidden we Create two new tensors with sizes n\_layers x batch\_size x hidden\_dim, initialized to zero, for hidden state and cell state of LSTM. Thus after creating the model we move it to the available device e.g: CPU, GPU based on the availability. After batching and loading as tensors we then move forward with the training phase of our project where After several trials and errors I have chosen 0.001 as the learning rate and 5 as epochs which could sometimes be adjusted. Inside the epoch we initialize the hidden state of our network, Calculate the loss in each epoch and we will do backpropagate through the entire training dataset. After all the epoch, We calculate the loss and performance and accuracy of our model and we use clip\_grad\_norm which helps our model to prevent gradient exploding. Finally, we get training, validation loss and training, validation accuracy after each epoch which will help us to create stopping criteria if the loss started to increase beyond certain level or if accuracy started to decrease after a certain point.

**Model Summary:**

           With the help of the crated model, we create a line graph between training and validation loss and another line graph between training accuracy and validation accuracy. After which, Using the prediction function, the model will be tested on unseen test data. Using sklearn modules such as accuracy\_score, classification\_report, confusion\_matrix and test target and predicted value we calculate the confusion matrix for the model and finally with accuracy score function accuracy of the model will be calculated.

**Improvement Suggestions:**

* The more context you can include, the better performance you will have so using a more advanced model like Bert and bidirectional lstm which captures the context in both ways. which will arguably increase the performance.
* And also doing hyperparameter search to optimize the configuration
* Using pre-trained word embeddings like Glove word embeddings
* Increasing the model complexity like adding more layers/ using bidirectional LSTMs

**Conclusion :**

           To conclude, In this project, we have seen how modern deep learning models are effective in text classification by using IMDB movie reviews. With this model, It is so simple to change the context completely such as tweeter sentiment analysis or emotion detection and the list goes on as the application of classification goes way beyond. To sum up, Our model has achieved an accuracy between 85 – 91% variably which is considered good but it again also can be improved by following very suggestions given above.

Note : Due to hardware capabilities of my laptop I used google colab to run this code, However, colab starts to crash when ram usage goes above certain limit so I switched my environment to Kaggle with ‘GPU’ turned on. For best results please try Kaggle.