TASK-1

Abstract—**This document consists of the write-up of coding Assignment of Sentiment Analysis and Machine Translation.**

I. Sentiment Analysis

A. *Loading The Comments adding the sentiments and Creating the Dataframe*

We are Developing a Simple Neural Network to do the sentiment analysis for this we have to download the datasets that contains the negative and positive comments and loading the positive and negative based on their file names and creating list to get those comments from the given files we are keeping and i tried to clean them but the accuracy of the RNN LSTM BILSTM GRU are less compared to the Simple neural network in the process of cleaning may be i had lost some information that can be useful for the training so i didnt clean them but i wrote them in comments. based on the names of those comments we are adding labels(sentiments) to those comments keeping both in same dataframe because we are training so they need to be in one dataframe.

B. *Data Division ,Tokenizer, Converting into Sequences and Padding*

We are going to spilt the data into Training(70) , Testing(15) , Validation(15) and creating a instance for tokenizer and tokenizer should be of basicenglish from this tokeinzer we are going to get the vocab size from train text only because the validation and testing data should not be seen.Based on the vocab size and we are going to replace the words with numbers in the vocabulary for Embedding purpose and the sentence length can also be different for diffrent comments so we are going the pad them with '0' and by taking max length based on their respective datasets

C. *Converting our Data into tensors and making Dataloaders of all the three*

We are converting our data into tensors because they can automatically compute the gradient of any differentiable expression making it highly suitable for machine learning.It can run not only on CPUs, but also on GPUs and TPUs, highly parallel hardware accelerators.And in our datframe our sentiment is positive and negative unlike zero or oneso we use labelencoder to modify them.And chnaging the comments and sentiments into tensors. In pytorch we have dataloader to efficiently handle the data it has has batchsize and shuffle as its arguments in this we create batch size so the optimization and everything handles in the form of batches if we pass more data ata time it does not optimize it and coming to hyperparameters embeddingdim = 200 hiddensize = 256 numclasses = 2learningrate = 0.001 num ofepochs = 20 embedding dimension is that as we are not using any pre defined word embedding slike word2vec and glove we use embedding layer in pytorch to create them and embedding dimensiuon is the size of that converted word embeddings and hiddensize is the size of the hidden state dimensions in this layer and num of classes is our positive and negative and regarding number of epochs is that how many time data is running again and again to optimize it learning rate the model learning speed it take only 0.001 steps to optimize it if we increase it we may not optimize it correctly.

D. *SimpleNNModel*

In the we define a class SimpleNNModel and create the neural network with what it should contain and input and output for every layer we use Relu activation function for applying non linearity to our data and also get some complex relations in the data.

E. *RNNModel*

we create a class with RNN and in RNN we will allow to carry information from one input to another input because it is applied sequential data only the model should remember the previous state memory in it so that is used to get the information of fututre and present data like time series data in stock market we use yesterday's value to get the todays stock value so want a memory unit that is hiddenstate . We inherit RNN in tensor we also have the predefined module of RNN so we apply that to our data.

F. *LSTM*

we create a class with LSTM and inherit the LSTM from the nn.module and the main diffrence in lstm and RNN is that in RNN we have only one hidden state that carries between previous to present but in lstm we have another cell that carries information from first input to last input if necessary based on three forget gate input gate and output gate.so it is named as Long-short memory network

G. *GRU*

we create a class with GRU and inherit the GRU from nn.module It combines the forget and input gates into a single update gate. It also merges the cell state and hidden state into same path and simpler than the LSTM

H. *BILSTM*

We create a class with BILSTM and inherit the BILSTM from nn.module. it is same as LSTM but it is bidirectional every input other than first and last will get the hiddenstates from past and present and also same with the memory cell and our problem is Three layer so we have three diffrent layers of

LSTM they combine the output from their below layers and give us the output.

I. *Training*

Train the models and evaluating them with same settings for all the five models.based on our accuracy we can say that because of smaller datasets our RNN LSTM BILSTM GRU performed not so well compared to the simple neuralnetwork.

II. Machine Translation

A. *Creating Inputs for Encoder and Decoder*

We Open our Text file and create two parts of it english sentences and spanish sentences we are doing this because those need to be send seperately for encoder and decoder in the transformer. we are adding start and end to our spanish sentences because it is a common preprocessing step in sequence-to-sequence models like machine translation to indicate the start and end of the sequence.

B. *Spilting the data into Train test val pairs*

We spilting the data into train test validation pairs with 70 15 15.

C. *Vectorizing the text data*

As we have to do the vectorization before passing it into the Embedding layer in the encoder and decoder we use TextVectorization from keras to do that and also we remove some punctation from the spanish texts and padding them to be of same sequence length and also we make datasets based on our batchsize which contains encoder and decoder inputs.

D. *Creating the transformer*

In transformer we will have Encoder and Decoder

*1) **Encoder**:* In encoder we will have the inputs as positional embedding to remember the positions of the words and after taking those inputs we will pass it to attention layer from the paper ”attention is all you need” i have used number of heads as 8 so we create 8 attention layers and concat them and Normalize them with layer Normalization and passing through feed forward neural networks same as our simple neural network to get some non linearity relation from the data and again normalize them pass those outputs in to decoder layer.

*2) **Decoder**:* Same as Encoder we pass positional embeddings and after first multihead attention we use the encoder output with decoder inputs and pass them into the second multihead attention get the attention weights and do the Normalization and feed forward to get the output but in the Decoder we use the output ans input for the decoder and decoder will take one at a time unlike the encoder it takes whole sentence at a time. Also we apply masking in decoder because we dont see the next word in the decoder so we apply masking.

*3) **Hyperparameters**:* Num of heads is to tell how many time we need to perform the attention in Multiheadlayer so that it passes 8 diffrent QUERY KEY AMD VALUE for that layer and applies softmax function to get the attention weights based on the formulae embedding dimension is nothing but the dimension to which vectorized input should be change into latent dimesnion is the dimension we used feed forward network as a hidden state dimension and drop rate is 0.5 to decrease the overfitting.

E. *Train and evaluate*

after creating all the necessary methods for transformer we train the model on the train dataset and validation dataset and use the transformer to translate our test data.