In The name of God

University Tehran

Engineering Facility

Electrical and Computer Engineering

Deep Learning with Applications

HW # 3

Amin Fadaeinedjad

810195442

Spring 99

Abstract

In this Computer Assignment we are going to work with Recurrent Neural Network also known as RNN's. At the first question of this assignment we are going to implement a 3 different type of networks for this task this considers a one directional LSTM, bidirectional LSTM and the pyramid LSTM. LSTM stands for long short term memory which has a number of non-linear units in itself. The first part we are going to use 1.6 million tweets got from Stanford university dataset and use it in order to recognize that if the tweet shows the sender a positive opinion or a negative opinion. In the first question we are going to use 3 different ways for this method. The following methods are directional, bidirectional and pyramid structure and therefore we are going to compare the result of the different methods with each other and in that case the accuracy and the loss will be measured and the confusion matrix of the following methods.

Question 1

At first we need to mention the pre-processing we have done on the data in order to use them, because the data that we are using are raw text from twitter and other information about the tweet and the sender therefore we need to get rid of the information that we don't need so it will reduce the memory we are using in this problem.

So we only kept *Label* and *Sentence* for this case others that didn't help us got removed, the next part we removed the '#' as the hashtag sign because I believed that the hashtag had meanings in it so deleting the complete word will have bad effect of the result so we used it in the tweet as a word, the next part where there was someone mentioned I believed that meant someone of a some page in twitter so we replaced it with the word '$someone$' so it will still have meaning and the structure of the sentence didn't change that much, we also got rid of the tweets that contained a link in them the reason was mentioned in the uploaded HW file and we also erased some characters that have no meaning that are in the code.

And we also changed the labels of the Train data and the Test data as it is mentioned the labels 4 are sign of positive, must be changed to 2, label 2 are sign of no sense tweet and must be changed to 1, label 0 are negative tweets and there is no need to change this labels.

In the next step we are using $glov42b$ dictionary for the word embedding this way for embedding needs a big dictionary for all the words so we used this dataset of words which contains a large number of words with and without meanings and they are all mapped in 300 dimension space.

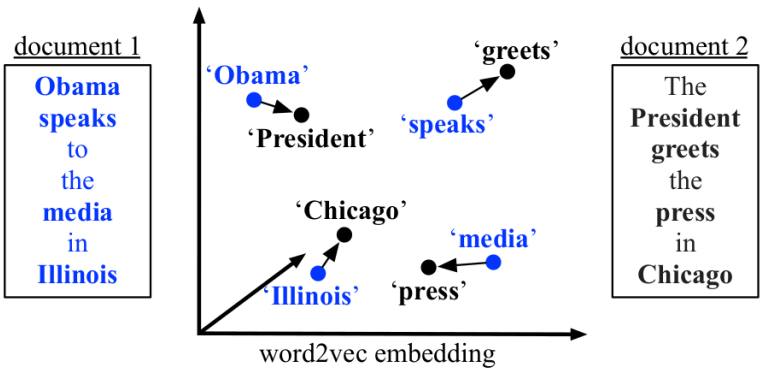


Figure 1

Figure 1 shows the vector space of 2 dimensional space, that the words are embedded the location of every word and we can see the distance of 2 words make sense that the words that have similarity with one another, according to the dictionary we are going to use 300 dimensional vector for every word.

The next part which is a very important part is the padding, we have several tweets and it contains a number of words and we need to pick a fix size for the dimension of the networks input therefore we need to pick a suitable size for the input, according to the HW file we need to pick the number of words in the tweet and that number is 280, but we used a code to calculate the size of the tweets and then we considered the maximum size of tweet the padding

size and for the number of data that we are using it's 40, the reason that we didn't use the 280 padding size is for the speed for the calculation therefore 40 is a good size.

We all of the following things have been done in the pre-processing stage of the work where we designed the data louder class that by passing the dictionary and the data frame to the class, so it can return the embedded vector as the result in the output.

In the next parts we are going to use the $LSTM$ (Long short term memory) for this word.

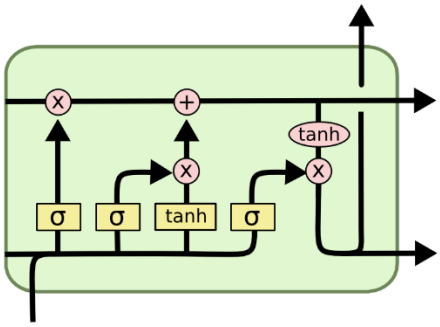


Figure 2

Figure 2 shows the structure of a simple LSTM and every element in the structure.

In this course we learned about different models such as RNN, LSTM and GRU which we are not going to discuss for now.

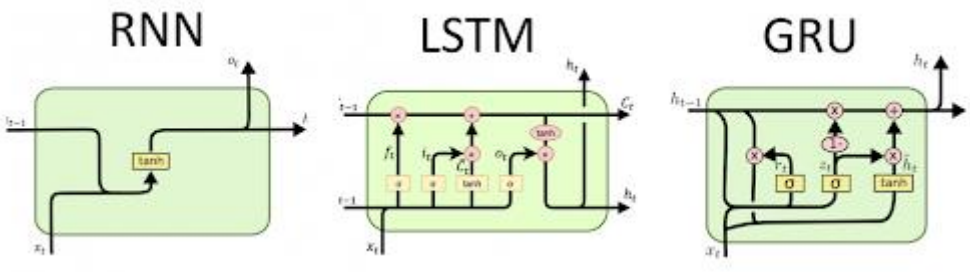


Figure 3

Figure 3 shows the difference between this 3 type of models and their structure.

1) In the first part we are going to use a single directional LSTM model for the classification of the tweet, at the first layer we are going to use an LSTM layer and in the output of that layer we are using a fully connected layer that has 150 neurons in the input and 3 in the output, which this 3 neurons are the score for every label that we mentioned before, and at the ending we are using a Softmax function in the last layer.

$$softmax(y_i) = \frac{e^{y_i}}{\sum_{j=0}^{n} e^{y_j}} \quad (1)$$

Equation 1 show the formula for the Softmax function.

It is needed to be mentioned for all the network in part 1 we assumed learning rate as 0.001 and the batch size of 16, and we used 5% of the train data as the validation data.

We used to cross entropy for the loss function for the classification.

$$loss(x, class) = -\log\left(\frac{e^{x[class]}}{\sum_j e^{x[j]}}\right) = -x[class] + \log\left(\sum_j e^{x[j]}\right) \quad (2)$$

Equation 2 is showing the formula for the cross entropy loss function.

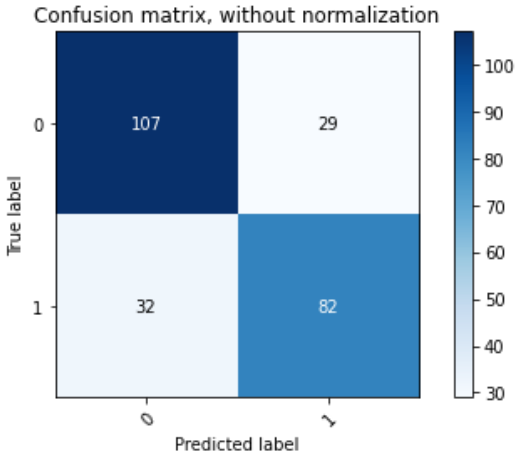


Figure 4 Validation

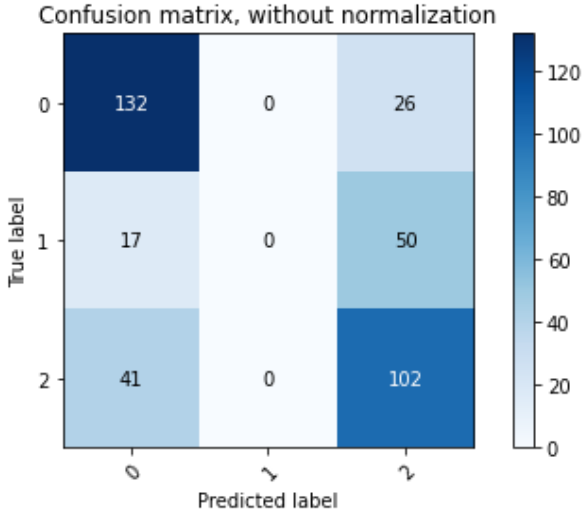


Figure 5 Test

Figure 4, 5 shows the confusion matrix for the validation data and the test data the figure 4 only has two labels because in that case in the test data we don't have 1 label there is only 0 and 2 as label so we just get this two labels in the confusion matrix.

In figure 5 we can see the result for the test data (needs to be mentioned that we cleaned the test data as well for the pre-processing), we can see that the predicted label never seems

to be 1 that's because of the training data that we are using and it never had seen label 1 as the result so in that case the network learns that it will never see and label 1 so that the reason it will never show.

After about 30 epoch of training of batch size 16 we got figure 6 and figure 7 as the result for the train loss and the train accuracy, it can be seen that the loss of the training data will decrease by time and for the train accuracy it will increase

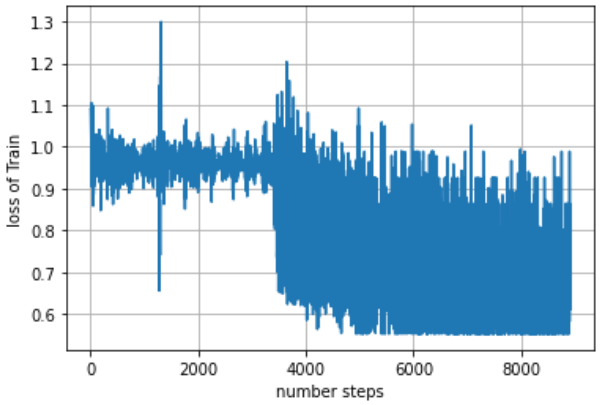


Figure 6 Train loss

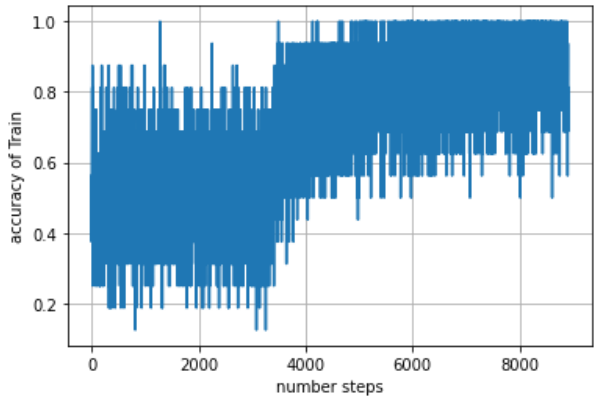


Figure 7 Train accuracy

It can easily be seen that the loss for Train will decrease more if we continue to train but we haven't been able to use GPU for the training so we have problems with the speed and can't do much more, but we still got a good result in the output.

The next part we also needed to plot the loss and the accuracy of the validation and test data but as it has been told we are using CPU so we can't just calculate the loss and accuracy for the two mentioned datasets in every step so we don't this for every 200 steps and we got about 41 points out of it and we plot the result and the following is the result for it

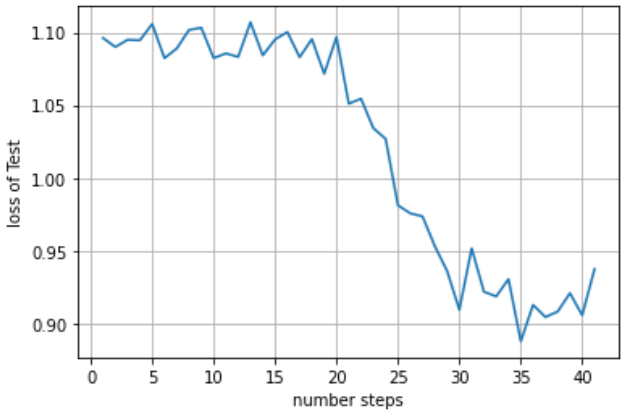


Figure 8 Test loss

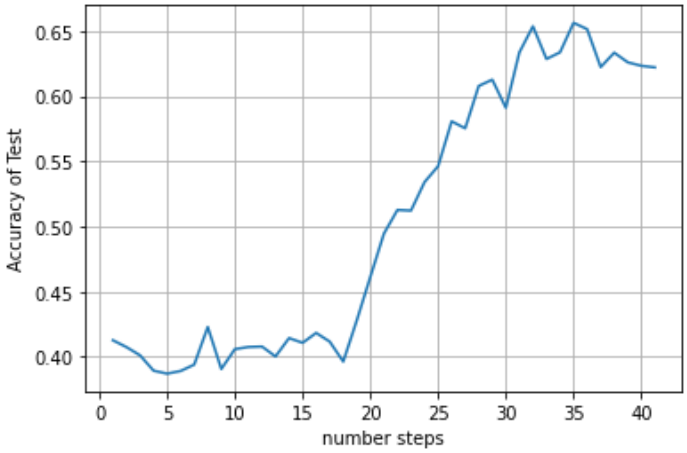


Figure 9 Test accuracy

Figure 8, 9 shows the loss and the accuracy for the Test dataset that we are using as you can see there is 41 steps for it and the reasons are mentioned. The next result we can find out is that after about 35 steps in the validation of the Test data, the loss starts to increase the same for the accuracy starts to decrease, we can find out that after step 35 the network starts to memorizing the data and in other words the network start to over fit the training data, the result for the test accuracy belongs to the final step of the computation.

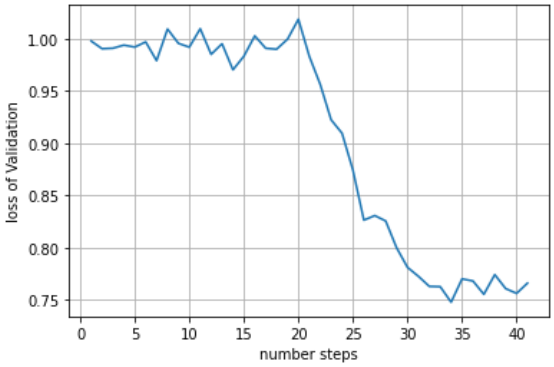


Figure 10 Validation loss

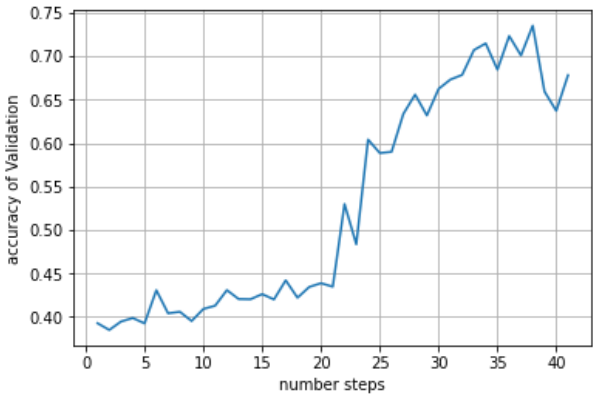


Figure 11 Validation accuracy

Figure 10, 11 shows the loss and the accuracy for the validation dataset the size of is about 250 tweets that in every step we are passed to see where we need to stop that calculations. Just like the test dataset we can't calculate the loss and the accuracy for every step instead like in the test dataset we measured it in every 200 steps. We are still getting the same results like the Test dataset but because the validation dataset doesn't have label 1 data so the loss is less than the test dataset and the accuracy is more than the one from the test dataset. So overall we are getting the same result but there is a small difference for these two datasets the reason it mainly because of the label 1 that is in the test dataset.

2) In the second part we are going to use a $Bidirectional$ model for the job, we can see figure 12 is the structure of a single directional LSTM model and figure 13 shows the structure of Bidirectional model, the bidirectional structure needs two time more parameters for the LSTM and in the output it will give us a twice size bigger output so in this case we need to do some changes in our code like setting the bidirectional passing argument to $nn.lstm$ as True and changing the size of h_0, c_0 the initiate values for the start and at last changing.

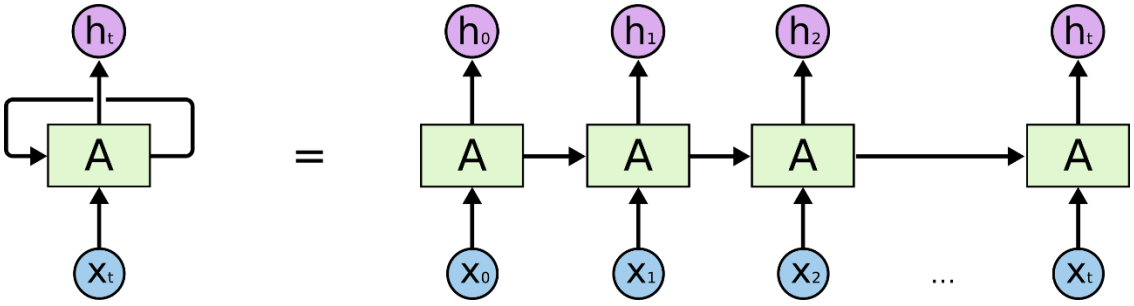


Figure 12

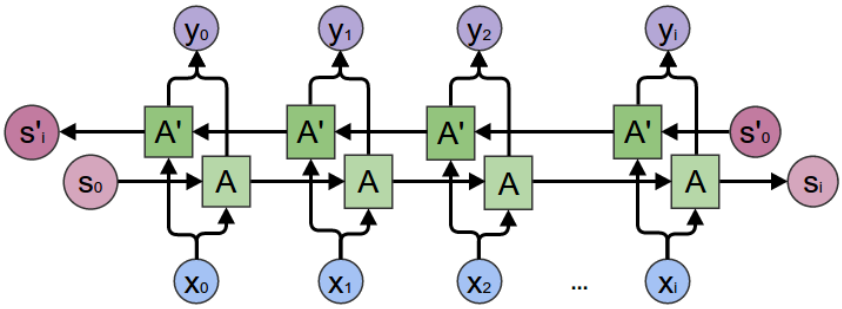


Figure 13

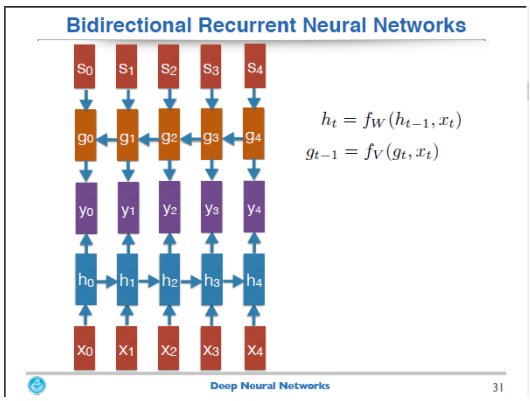


Figure 14

Figure 14 is a sample of how the Bidirectional RNN look like as RNN model, this image has been taken from the course slides Therefore, this LSTM models work just like the single directional but the difference between the bidirectional and single directional LSTM is that the first outputs of the single direction LSTM only depend on the first inputs and don't really matter about the others inputs after that input, but in bidirectional model every output is a function to all the inputs and changing any of the inputs will put a result on the output and this is better that the single directional model because we can't really find out if a message is positive of negative unless we read to the end of the sentence so the bidirectional model will do better in the same number of iteration the result we got is from a smaller number of epochs but for a same number of epochs of training the bidirectional has a better chance for its performance over the dataset.

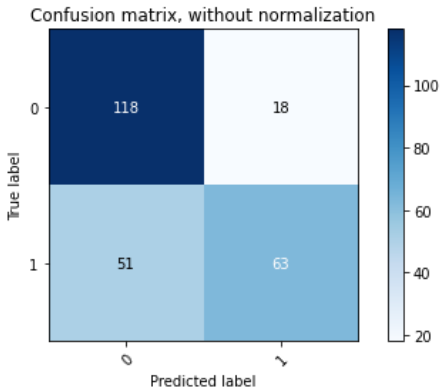


Figure 15 Validation

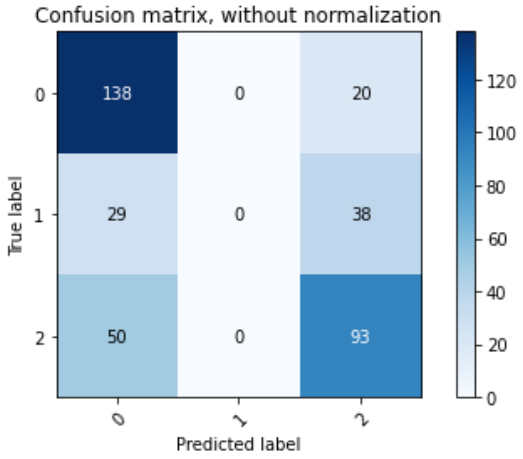


Figure 16 Test

Figure 15, 16 shows the confusion matrix for the test data and the validation data so we can see the accuracy of the test dataset and the validation dataset and at had been mentioned before the reason that we didn't predict label 1 is that because it hasn't seen the label 1 ever before so we will not see that in the predicted labels.

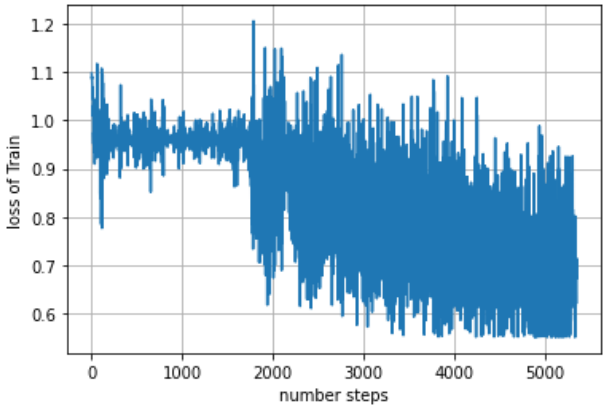


Figure 17 Train loss

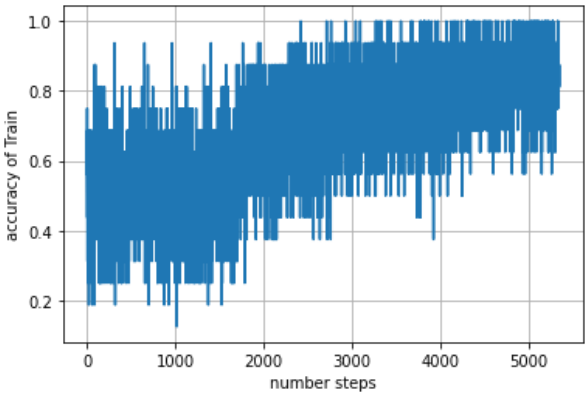


Figure 18 Train accuracy

In figure 17, 18 we can see the loss and the accuracy of the training dataset, as it can be understood from the figures the loss will start to decrease and the accuracy will start to

increase as it can be predicted, in the end of training data we can see the accuracy is getting close to 100% in some batches so there is a chance for the overfitting in the results.

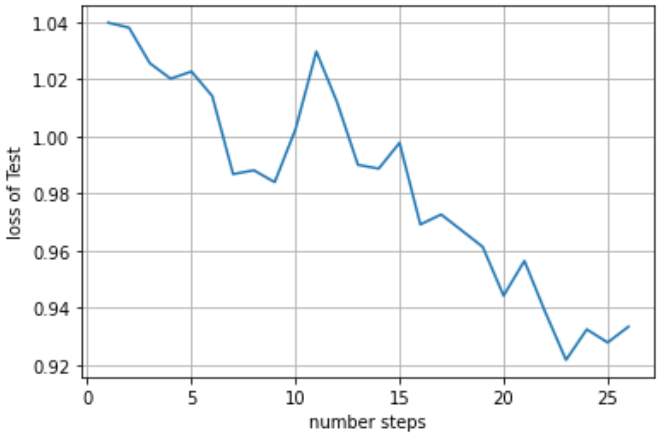


Figure 19 Test loss

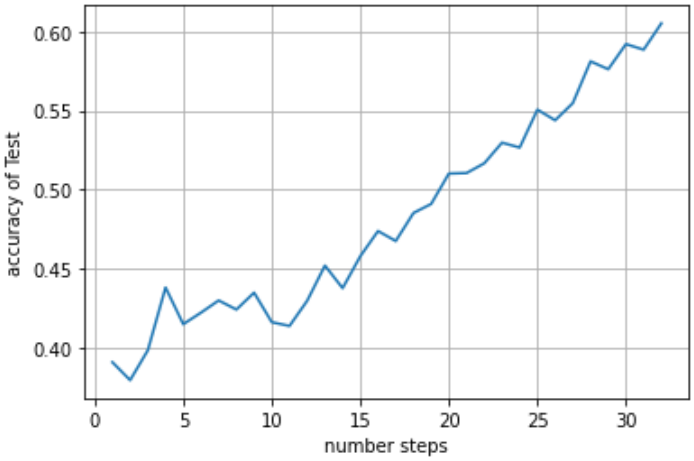


Figure 20 Test accuracy

Figure 19, 20 shows the loss and the accuracy for the test dataset and it can be seen the loss is decreasing by time and the accuracy is increasing there is no usual pattern for the loss and the accuracy it has a term changes randomly if we were able to do this for a longer time then it might have reached state where the loss was less and the accuracy was more than the thing that we reached right now but because of the CPU and memory that we are using we are not able to do this for a large number of epochs, but still the results that we got are reasonable as an answer.

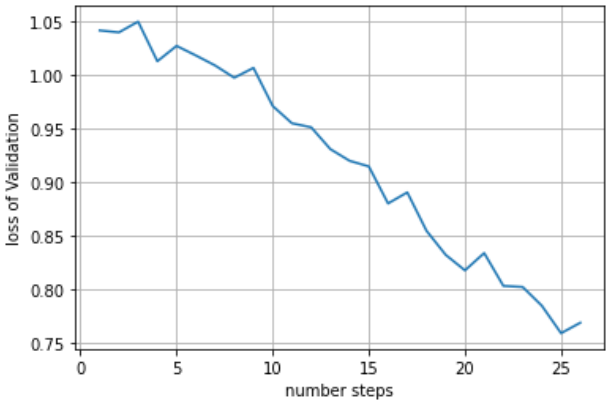


Figure 21 Validation loss

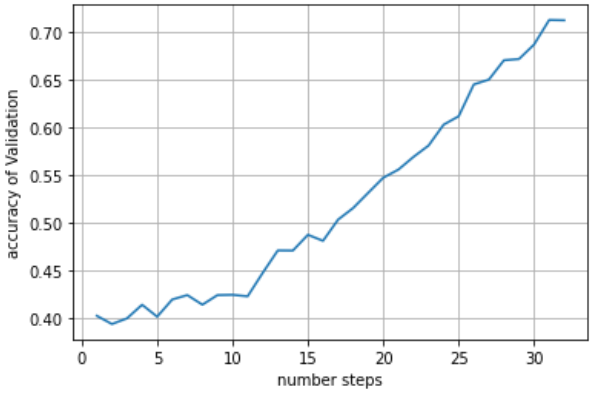
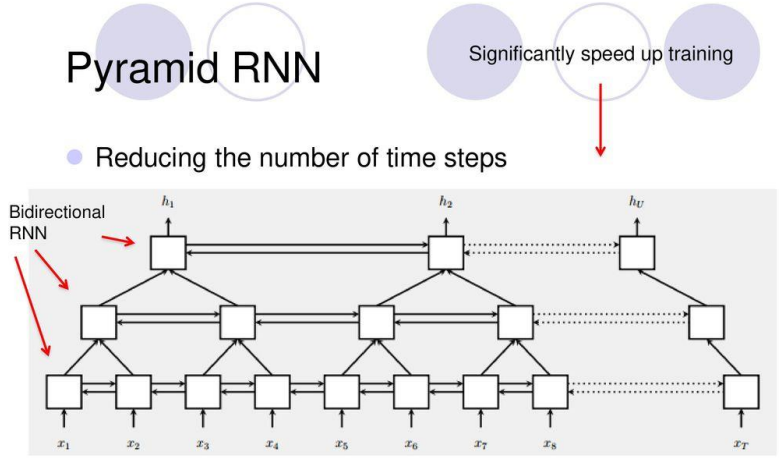


Figure 22 Validation accuracy

Figure 21, 22 shows the loss and the accuracy for the validation dataset and it can be seen the loss is decreasing by time and the accuracy is increasing there is no usual pattern for the loss and the accuracy it has a term changes randomly if we were able to do this for a longer time then it might have reached state where the loss was less and the accuracy was more than the thing that we reached right now but because of the CPU and memory that we are using we are not able to do this for a large number of epochs, but still the results that we got are reasonable as an answer. The result of the validation dataset is a bit better than the test dataset because the test dataset contains some tweets that their labels are 1 and that isn't in our train dataset so we can't expect to get and outputs for the reasons that were mentioned.

3) In the last part we are going to use the pyramid structure for out classification we can see in the figure 23.



W. Chan, N. Jaitly, Q. Le and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," ICASSP, 2016

Figure 23

We way that the *pyramid RNN* works is based on concatenation which we concatenate the lower level *LSTM* models, two by two and send the result to the higher level. This structure can do better because of the architect that it has, in this part we are going to use a 4 layer LSTM where the first hidden layer size is 64, and in every step we doubled the output size of the layer size so the next layer will have to get the input size of 128 because we are going to concatenate the two lower layer of the model so in this case because we have chosen the padding size be 40 in every level the input size the second dimension with reduce to its half size so in next layer the second dimension of the input will be 20 and the next layer will be 10 and the next will be 5.

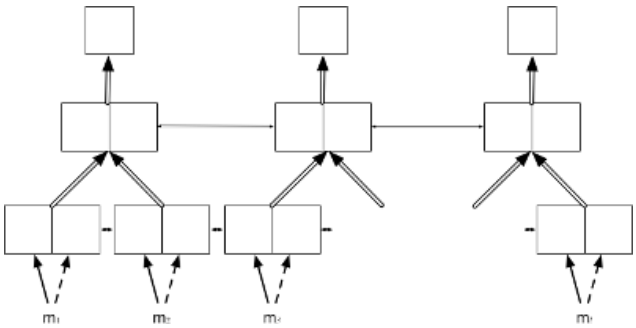


Figure 24

Figure 24 shows the result of concatenation in every layer that we combine two hidden layers a feed them to the next layer input. In every layer of the network we have the input size of that layer will change its size and becomes twice as big as it was before and in the last layer we have size of 512 and we feed it to a fully connected layer and get 3 scores in the output a by using a Softmax and loss function we train this network.

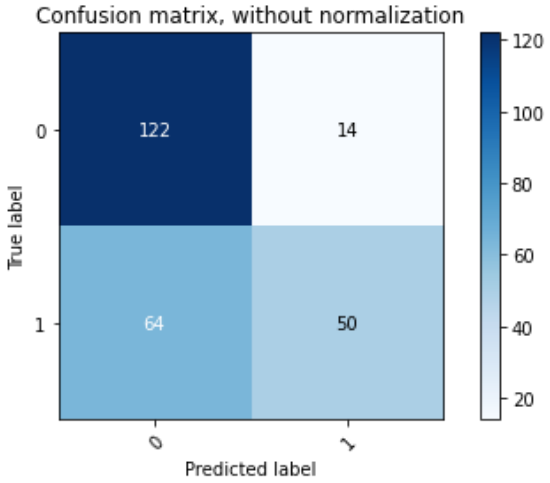


Figure 25 Validation

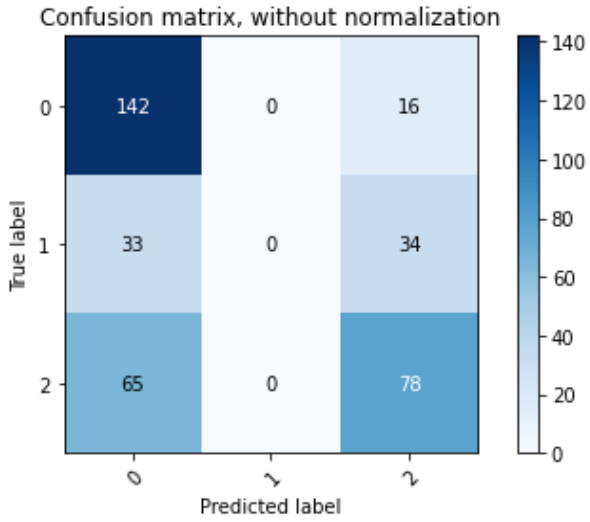


Figure 26 Test

Figure 25, 26 shows the confusion matrix for the test data and the validation data so we can see the accuracy of the test dataset and the validation dataset and at had been mentioned before the reason that we didn't predict label 1 is that because it hasn't seen the label 1 ever before so we will not see that in the predicted labels.

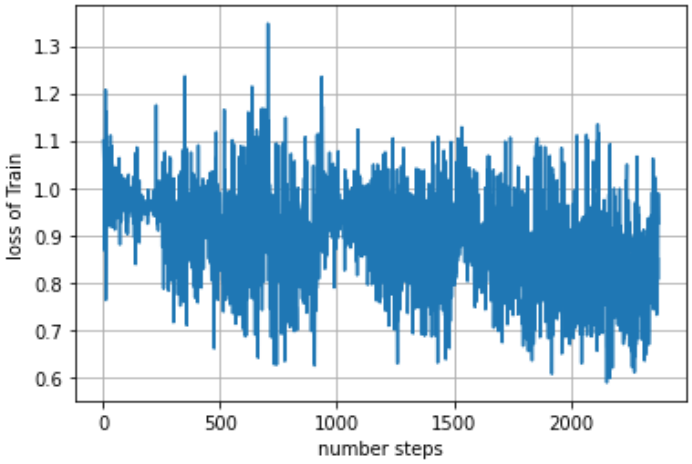


Figure 27 Train loss

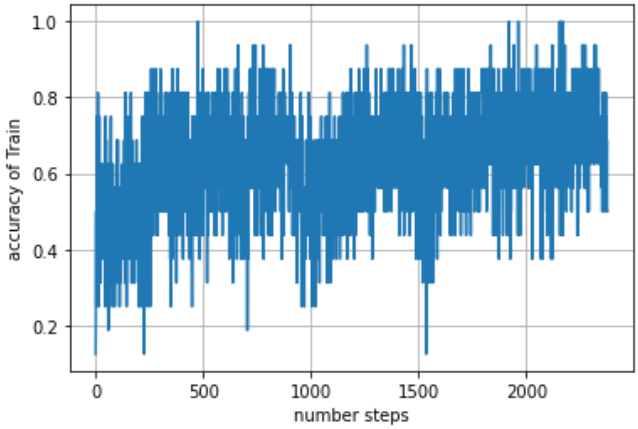


Figure 28 Train accuracy

Figure 27 and 28 show the loss and the accuracy of the model during the training, it can we seen that the average of the loss is decreasing during time and the accuracy is increasing during this time but we can still see some undershoot and overshoot in accuracy and the loss but overall it is doing a good performance but because we are using CPU and not the GPU so we can't do this training for a long time but still we got reasonable result at it because if we were able to do more training we might have been able to get better accuracy and lower loss, so as the result I believe the pyramid model might have better performance compared to the other two methods single directional and bidirectional model.

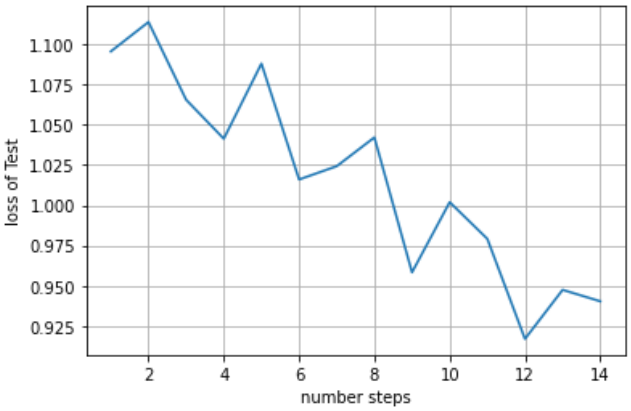


Figure 29 Test loss

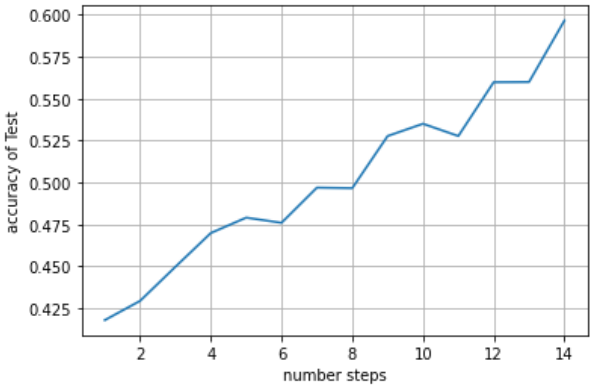


Figure 30 Test accuracy

Figure 29, 30 shows the loss and the accuracy for the test dataset and it can be seen the loss is decreasing by time and the accuracy is increasing there is no usual pattern for the loss and the accuracy it has a term changes randomly if we were able to do this for a longer time then it might have reached state where the loss was less and the accuracy was more than the thing that we reached right now but because of the CPU and memory that we are using we are not able to do this for a large number of epochs, but still the results that we got are reasonable as an answer.

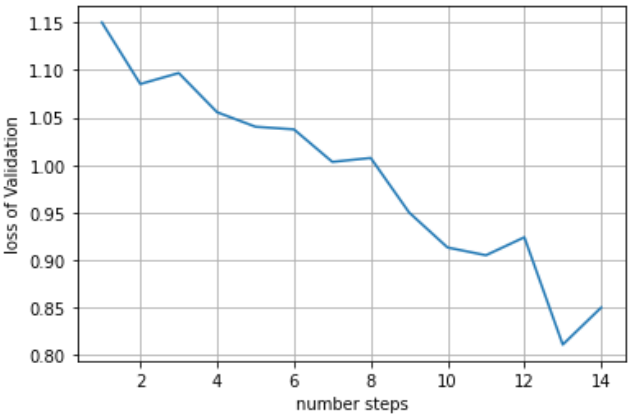


Figure 31 Validation loss

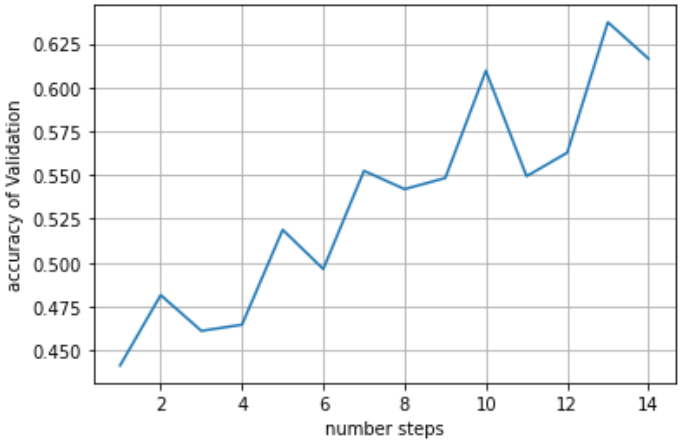


Figure 32 Validation accuracy

Figure 31, 32 shows the loss and the accuracy for the validation dataset and it can be seen the loss is decreasing by time and the accuracy is increasing there is no usual pattern for the loss and the accuracy it has a term changes randomly if we were able to do this for a longer time then it might have reached state where the loss was less and the accuracy was more than the thing that we reached right now but because of the CPU and memory that we are using we are not able to do this for a large number of epochs, but still the results that we got are reasonable as an answer. The result of the validation dataset is a bit better than the test dataset because the test dataset contains some tweets that their labels are 1 and that isn't in our train dataset so we can't expect to get and outputs for the reasons that were mentioned.

The table below show the loss and the accuracy of the test data after a number of epochs at has been mentioned we had problems connecting the GPU and had memory problems with Colab so we weren't able to run the training model for a long time but as the result that we saw in the graph we predict that the pyramid model has a notable performance, and the bidirectional has a faster way to get to the result.

Method	Number Epoch	Loss	Accuracy
Single Direction	30	0.9122	64%
Bidirectional	18	0.9229	62.78%
Pyramid	8	0.9415	59.78%

Question 2

In this part we are going to use the BERT model for our classification and the model is already pre-trained so the initial values are suitable for the job therefore we are using the previous data for training the model, and the only thing that we need to do it try to feed out own data to the network and see what will happen, because the BERT model is pre-trained and the fully connected layer isn't pre-trained so in that case we need to define a different learning rate for it. In this part we assumed the batch size of 32 and the dataset that we used 0.00001 as the learning rate for the pre-trained part and 0.01 for the fully connected part. We also after 200 iteration got a test on validation data and the test data to see the loss and the accuracy of we got, because we have 10000 tweets as the train data 5% of it is 500 tweet are for the validation dataset.

The tokenizer in the Pytorch library adds two tokens to the beginning and the ending of the tokens, and there is a mask that works with attention and shows that which of the tokens are used in it so the ones that are not used will be zero. The tokenizer did the same job in past part as the word embedding that gave us a token to work with instead of string.

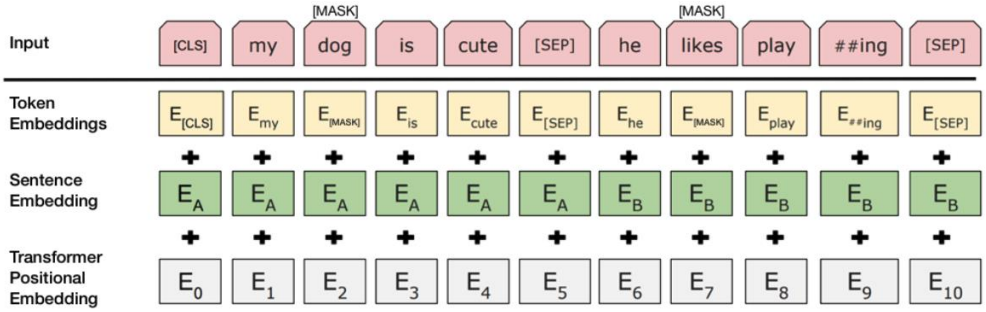


Figure 33

Figure 33 shows the structure of the BERT model and how it works in every stage of itself.

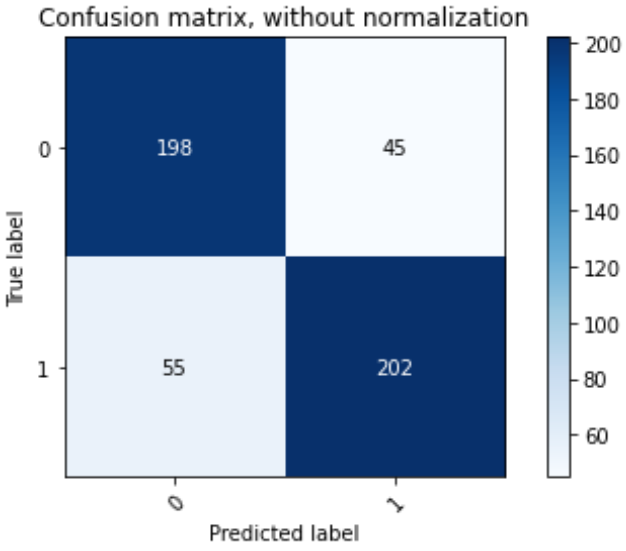


Figure 34 Validation

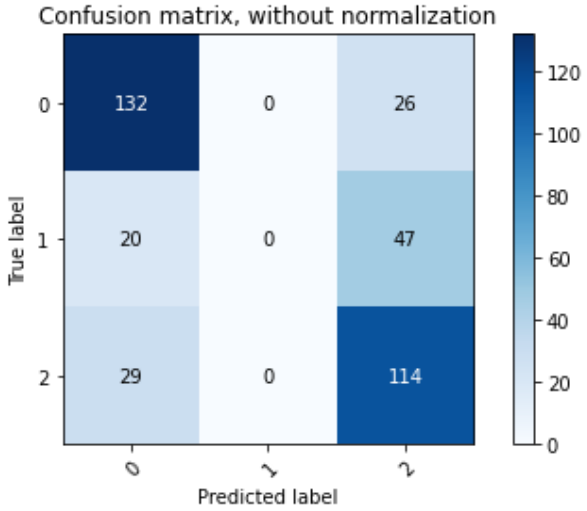


Figure 35 Test

Figure 34 and 35 shows the confusion matrix of the validation and test data we have, the accuracy for the validation dataset if 80% while the accuracy for the test dataset is 66.85% so there is a gap between the two accuracy that we have in the validation dataset and the test dataset, as it has been mentioned in the previous parts the reason that we haven't seen a label 1 in the predicted labels is that the training data that we are using doesn't have any of that label so it will become a kind of memorization that there is no label.

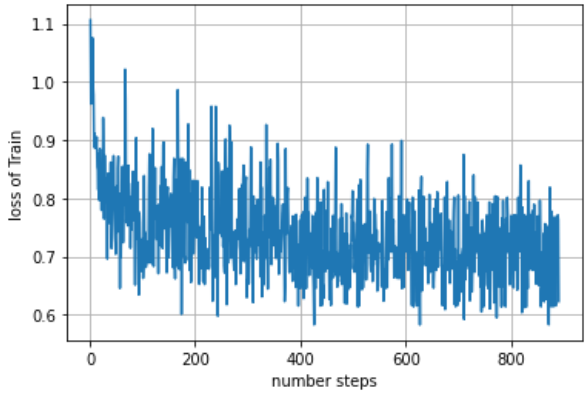


Figure 36 Train Loss

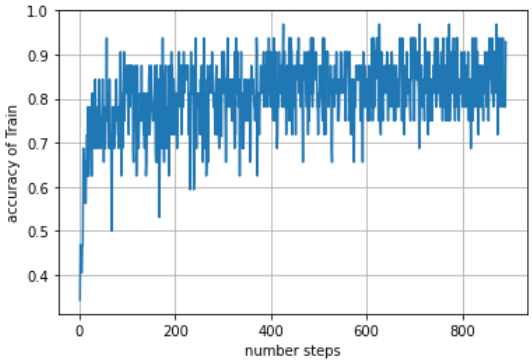


Figure 37 Train Accuracy

In figure 36, 37 we can see the loss and the accuracy of the training dataset, as it can be understood from the figures the loss will start to decrease and the accuracy will start to increase as it can be predicted, and we can see where this is going to.

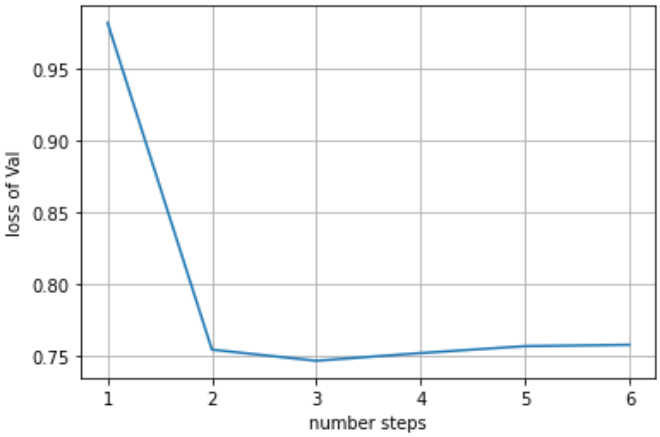


Figure 38 Validation Loss

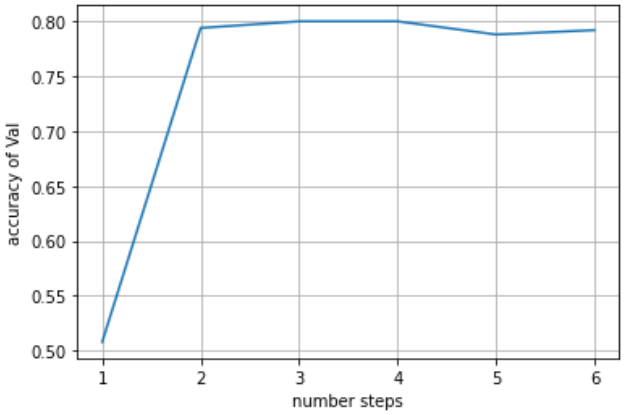


Figure 39 Validation Accuracy

Figure 38, 39 shows the loss and the accuracy of the validation data, as we are watching the graph of the loss and accuracy we can see that at the beginning the loss starts to decrease and the accuracy will start to increase but isn't what it happens to end, because we are memorizing the dataset from a step beyond, we are looking a good to be able generalize the model to other datasets so the best place the model is where the validation loss starts to increase of the validation accuracy start to decrease.

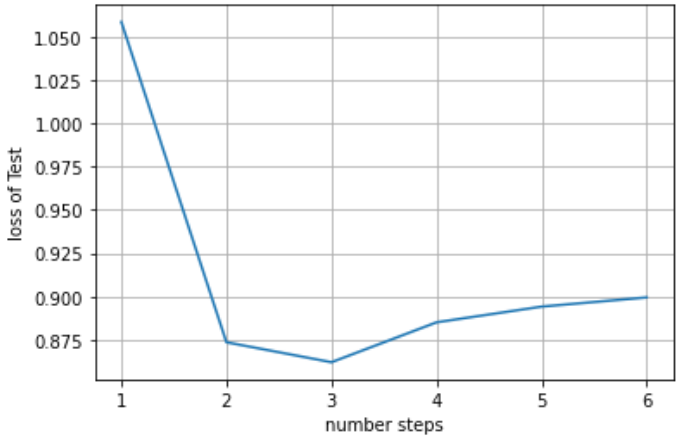


Figure 40 Test Loss

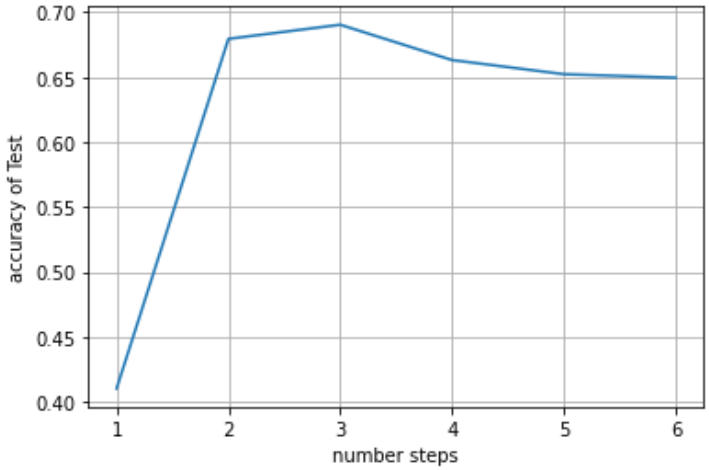


Figure 41 Test Accuracy

Figure 40 and 41 shows the loss and the accuracy of the test dataset that we have just like the validation dataset the test dataset shows a similar result to some of its behavior like we can see that at first the loss starts to decrease and the accuracy starts to increase but after a certain number of epochs we consider that the network as memorized the dataset and to its performance for generalizing will decrease so in that case the loss will start so increase and the accuracy will start to decrease. The reason that as always the performance for the validation dataset in more valid than the others is that we only have data with label 0 2 in the validation dataset just like the training dataset but the test dataset has label 1 and that will cause some miss classification because of the label that we never have sawn in the output might be seen in the output.

Process

In this Computer assignment we learned about Recurrent Neural Network and the way that we use them in order to classify the label of the sentence we also saw different architects of the RNN model like single directional bidirectional, pyramid and we compared the speed of learning and the speed of forward pass of the networks and got different results for them all we could have found out that the pyramid has a long time training but in be trained in smaller number of epochs.

The last part we used a pre-trained model to classify the tweets mentioned and it had a better performance compared to the previous models like single directional and … the reason that might the BERT model work well is that the first layer that we used already have had a good value so there was no need to change that much instead we worded on the fully connected layer more.

Reference

https://towardsdatascience.com/bert-classifier-just-another-pytorch-model-881b3cf05784

https://colab.research.google.com/drive/1pTuQhug6Dhl9XalKB0zUGf4FIdYFlpcX

https://medium.com/swlh/painless-fine-tuning-of-bert-in-pytorch-b91c14912caa

https://discuss.pytorch.org/t/understanding-output-of-lstm/12320/2?u=jpeg729

https://www.reddit.com/r/deeplearning/comments/9lhnuh/questions_about_lstm_and_pytorch/

https://discuss.pytorch.org/t/the-difference-and-use-of-output-and-hidden-state-of-an-rnn/15108

https://stackoverflow.com/questions/48302810/whats-the-difference-between-hidden-and-output-in-pytorch-lstm

https://stackoverflow.com/questions/56506412/pytorch-lstm-input-dimension

https://stackoverflow.com/questions/45022734/understanding-a-simple-lstm-pytorch

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

http://help.sentiment140.com/for-students

https://pypi.org/project/pytorch-pretrained-bert/

https://towardsdatascience.com/bert-classifier-just-another-pytorch-model-881b3cf05784

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Listen, Attend and Spell, William Chan, Navdeep Jailty, Quoc V.Le, Oriol Vinyals