Assignment-2 Report: Sentiment Analysis in NLP using RNNs

Abhishek Tanpure - 17D070018
Anwesh Mohanty - 170070009
Rohan Bansal - 170070058

1 Word Embeddings

We have imported a pre-trained word embedding glove model to show the behaviour on some analogy tasks. We have tried for 2 different dimensions 50 and 300 using the glove model having 6 Billion tokens and 400k Vocab size. We have calculated the L2 norm between word embeddings, however to give more insight into predictions with have also used top 10 nearest neighbours for a word embedding using knn search. Consider the 5 examples shown below. L2 distances are shown below for the 50 dimension glove model.
1) 'man' - 'woman' + 'son' = 'daughter'
2) 'beijing' - 'china' + 'tokyo' = 'japan'
3) 'bad' - 'worst' + 'big' = 'biggest'
4) 'do' - 'did' + 'go' = 'went'
5) 'prince' - 'boy' + 'girl' = 'princess'
6) 'king' - 'male' + 'female' = 'queen'

L2 dis between 'man' - 'woman' + 'son' and 'daughter' is 1.58.
L2 dis between 'beijing' - 'china' + 'tokyo' and 'japan' is 2.78
L2 dis between 'bad' - 'worst' + 'big' and 'biggest' is 3.48
L2 dis between 'do', 'did', 'go' and 'went' is 1.79
L2 dis between 'prince' - 'boy' + 'girl' and 'princess' is 10.5
L2 dis between 'king' - 'male' + 'female' and 'queen' is 11.54
Following are the analogy tables for each of 50 and 300 dimension models:

Example No.	1)	2)	3)	4)	5)	6)
KNN 1	daughter	japan	biggest	went	girl	female
KNN 2	mother	tokyo	worst	before	boy	male
KNN 3	wife	japanese	big	came	kid	bisexual
KNN 4	son	singapore	sweep	when	toddler	adults
KNN 5	niece	shanghai	nation	took	teen	adult

Table 1: 50 dimensions Glove model KNNs for each example

Example No.	1)	2)	3)	4)	5)	6)
KNN 1	daughter	tokyo	worst	went	girl	female
KNN 2	son	japan	biggest	go	boy	male
KNN 3	mother	japanese	big	did	girls	males
KNN 4	wife	yen	sweep	came	teenager	females
KNN 5	eldest	asia	ever	gone	boys	women

Table 2: 300 dimensions Glove model KNNs for each example

It is surprising but one can clearly see that the 50 dimensions glove model is performing better than the 300 dimension model. This is apparent in the 2) and 3) analogy tasks where the model should have had the first nearest neighbour as 'Japan' and 'biggest' respectively as is the case in 50 dimension glove model however that is not true for the 300 dimension glove model. Both the models fail to do well on analogy tasks 5) and 6), instead both of them output the same nearest neighbour which does not match with the desired output.

2 Sentiment Classification

We build a sentiment classifier using a recurrent neural network framework (either LSTM or GRU or some variation of them) in PyTorch. The standard framework for our experiments is:

- An embedding layer to store the different embeddings of the words encountered in the reviews. We have used both pre-trained embeddings (from Glove) and trained our own embedding layer to observe how much the output varies.
- A RNN layer. We have either kept this RNN layer as a LSTM or GRU (with an option to turn on bidirectionality). We have varied the number of hidden layers, number of units stacked together and bidirectionality function to check the effect on the final output.
- The final layer is a linear layer which provides a single output to denote the sentiment in the input review.
- There is also a dropoout added in the LSTM and linear layers.

There are more than 500000 distinct words in the reviews but we could not set the embedding layer input to the total number of distinct words due to constraints on computing power. Hence we make a vocabulary using the most frequent 50000 words present in the reviews.

3 Hyperparameter tuning

In this section, we try to improve upon the accuracy obtained previously. For this we have identified 6 different components where there might be a scope of improvement, and present

an extensive study of the behaviour of the models when one or more of these components are varied. The components/hyperparameters identified are:

3.1 Different Number of Units

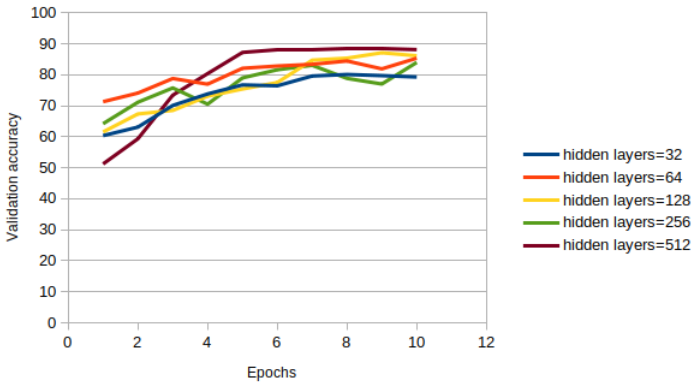

Figure 1: Different number of units

Here we manipulate the number of hidden layers present inside the LSTM. As is expected generally, for larger number of hidden layers the performance is a bit on the lower side during the first 2-3 epochs but then quickly rises to the top accuracy as can be seen fo hidden layers=512 (brown line). For lower number of hidden units, the initial accuracy might be higher than others but the final accuracy is lower generally when compared to higher values of the parameter as the model is not able to encode the sequential data better compared to models with higher hidden units. The default dropout here is 0.5 and number of lstm layers are set as 2.

3.2 Additional LSTM layers

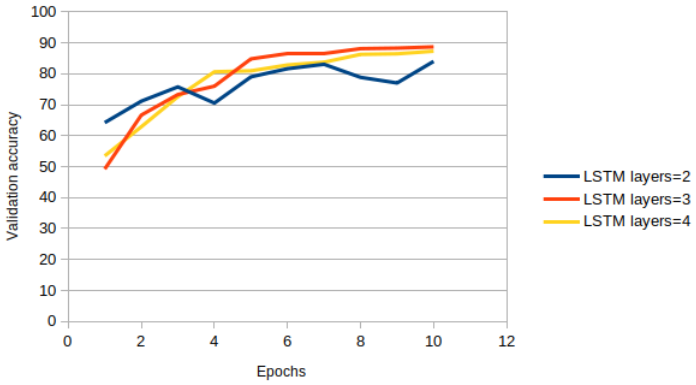

Figure 2: LSTM layers

Here we change then number of lstm layers stacked together. The accuracy for 3 lstms and 4 lstms is pretty close but both of them perform much better than the 2 lstm layer case. It might be possible that there is a small amount of overfitting happening for the 4 lstm layer case due to which its accuracy is slightly lower or maybe it hasn't gotten enough time to reach its maximum accuracy given the frame of only 10 epochs. For 2 layers of lstm it is possible that the model is unable to learn the sequential data well enough resulting in the comparatively poor performance. The default dropout is set as 0.5 and the number of hidden units is set as 128.

3.3 BiLSTM (BONUS)

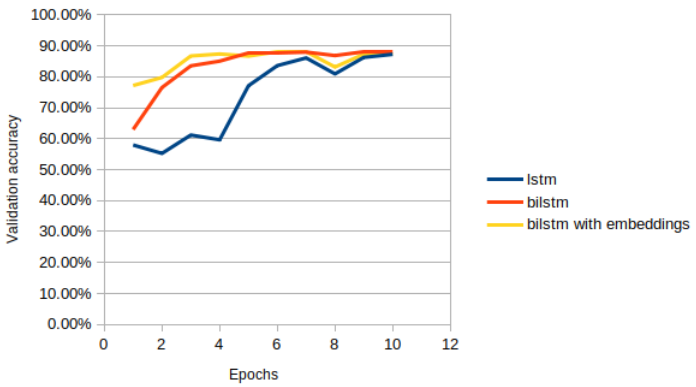

Figure 3: Using BiLSTM

This is for the **bonus** part. We enable the bidirectional attribute of the LSTM. It is clearly visible that the BiLSTM is able to capture the sentiment of the sentence in the dataset much faster than the LSTM. Also it gives a comparatively greater accuracy in the initial stages as well as a slightly better accuracy in the later epochs as well. Using the Glove pre-trained embeddings further accelerates the training process. The other parameters are set as- dropout=0.5, number of layers=2, hidden units=128.

3.4 Replacing LSTM with GRU

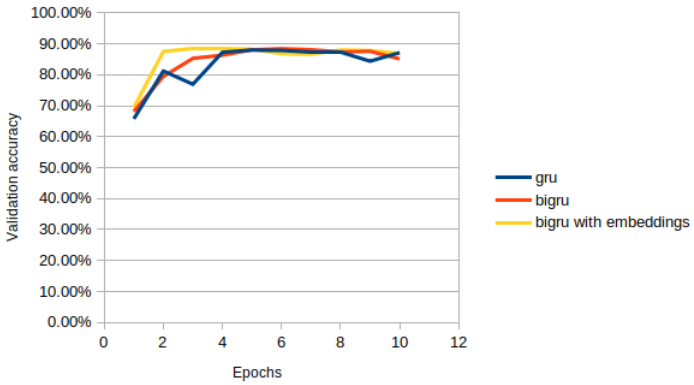

Figure 4: Using GRU

In this section we replace the LSTM unit by a GRU unit. The training curve observed for the GRU is similar to the LSTM curves but their is one significant difference. The average epoch time for the LSTM was around 13s but for a GRU was close to 9s keeping the other hyperparameters as constant. For larger models/epochs this will lead to much lesser times without compromising much on the accuracy (or even better accuracy). The other parameters are set as- dropout=0.5, number of layers=2, hidden units=128.
Bonus: We have also included the training curve obtained after using the BiGRU and BiGRU+pre-trained embeddings model. The general trend observed is similar to the ones obtained in the previous part with the BiLSTM. Using a BiGRU accelerates the training process and using pre-trained embeddings with it makes it even more faster. The final accuracies obtained are similar to the ones obtaiened using BiLSTMs. Hence BiGRU + pre trained embedding model gives the best accuracy and is much faster than other models which give similar accuracies.

3.5 Dropout

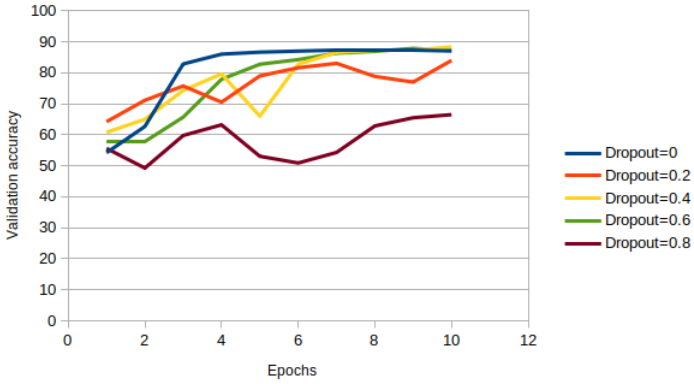

Figure 5: Different dropouts

Best accuracy is observed for dropout in range 0 to 0.4. Too much dropout(=0.8) and we start to lose necessary information which was expected since 8 out of the 10 times the weights were getting dropped which is too high a probability for the model to learn effectively. Although not clearly visible in the graph, the dropout accuracy on the validation data is highest after 10 epochs for a dropout value of 0.4

3.6 Weight decay

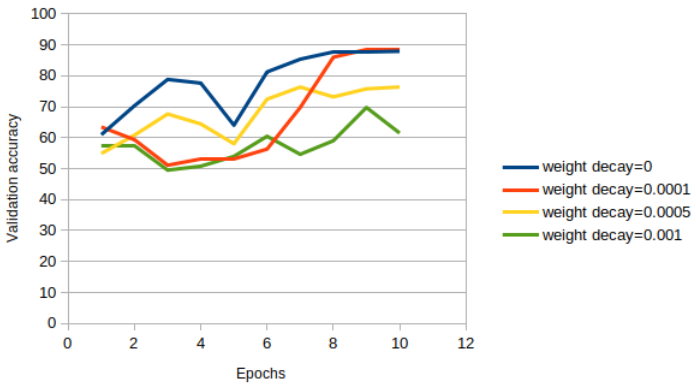

Figure 6: Weight Decay

One can observe that the best performance is obtained for a weight-decay of 0. For each epochs the validation accuracy is better in that case. Although one cannot observe a specific trend in the results for varying weight decay, we can that to reach a high enough accuracy we must have weight decay as low as possible.

4 Test on another dataset

We have used the stanford sentiment treebank (SST) dataset to test our trained model. SST handles the crucial task of sentiment analysis in which models must analyze the sentiment of a text. To test the model we have used models trained using hyperparameters that provide the best performance. To have a more clearer insight into the testing process we have also plotted the graph to test accuracy with each training epoch. NOTE: Here the training is being done on the original IMDB dataset since the objective here is to see whether the trained model is able to generalize well on other dataset.

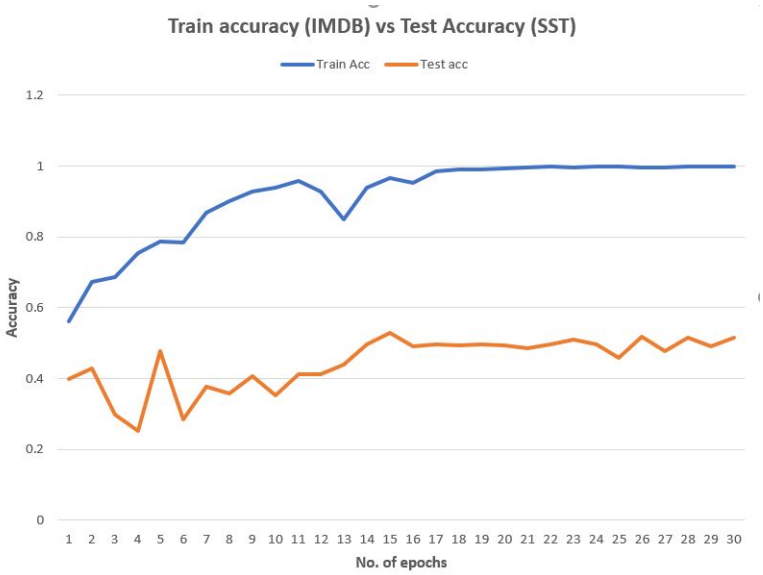

Figure 7: Weight Decay

The train accuracy, val accuracy and test accuracy on the other dataset are shown below.
Train Loss: 0.003 — Train Acc: 99.97
Val. Loss: 1.098 — Val. Acc: 86.06
Test Loss: 2.984 — Test Acc: 51.44
From this one can clearly infer that the model is overfitting to the dataset since the train and validation accuracy are high whereas the test accuracy (data from other dataset) is very low. This can happen when it comes to NLP tasks since they learn the type of language used from the first dataset, however when there is a change in the semantics the model fails to generalize.

5 Bonus

For the bonus part, we have chosen to work with a BiLSTM model. In addition to this, we present an analysis of the performance on the dataset by using a BiGRU model as well as by including pre-trained word embeddings + packed RNN sequences inside the LSTM/GRU layer. The inferences and plots for the bonus models have already been covered in Sections 3.3 and 3.4.

6 References

Provided in the codes submitted .