

Name – Bbiswabasu Roy

Roll – 19EC39007

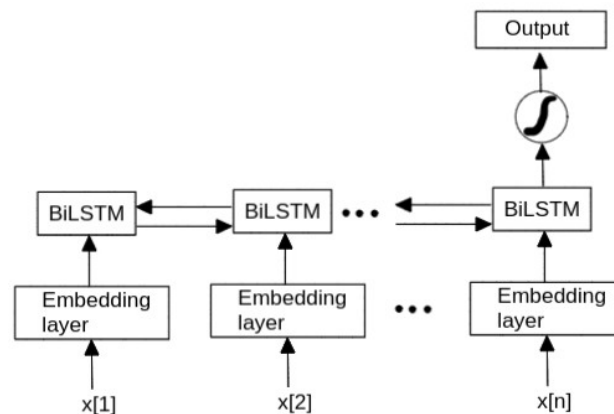
Assignment – 2

Objective:

Given a labelled dataset of movie reviews along with the sentiments, i.e, whether they are positive or negative, design a BiLSTM model to predict the sentiments of test examples.

Design:

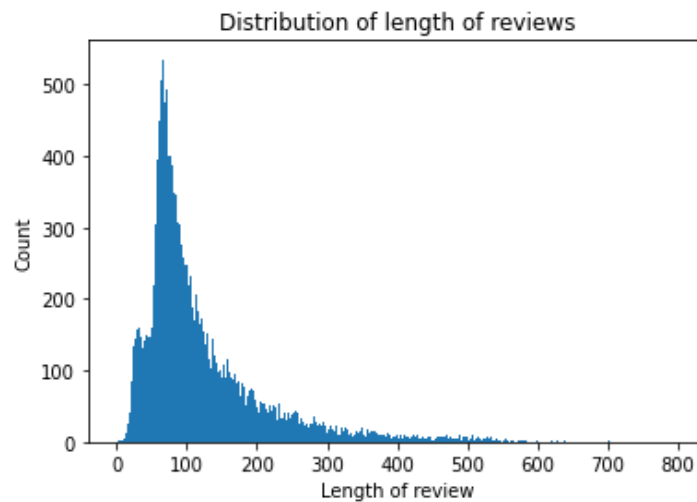
- The reviews in the dataset were first preprocessed by removing HTML tags, square brackets and special characters
- Vocabulary was formed using reviews present in training set and each word was assigned an id. All the reviews were converted to sequence of tokens and the labels were converted to numerical values, i.e, “positive” was converted to 1 and “negative” was converted to 0
- Pretrained GloVe vector was loaded and the embedding for each word was stored in *weights_matrix*
- The architecture of the neural network is as follows:



- Binary Cross Entropy loss was chosen as the loss function
- Hyper-parameters were chosen appropriately with some fine tuning so as to get good results. For tuning the hyper-parameters, the F1 score of the model on the validation set was evaluated.

Results:

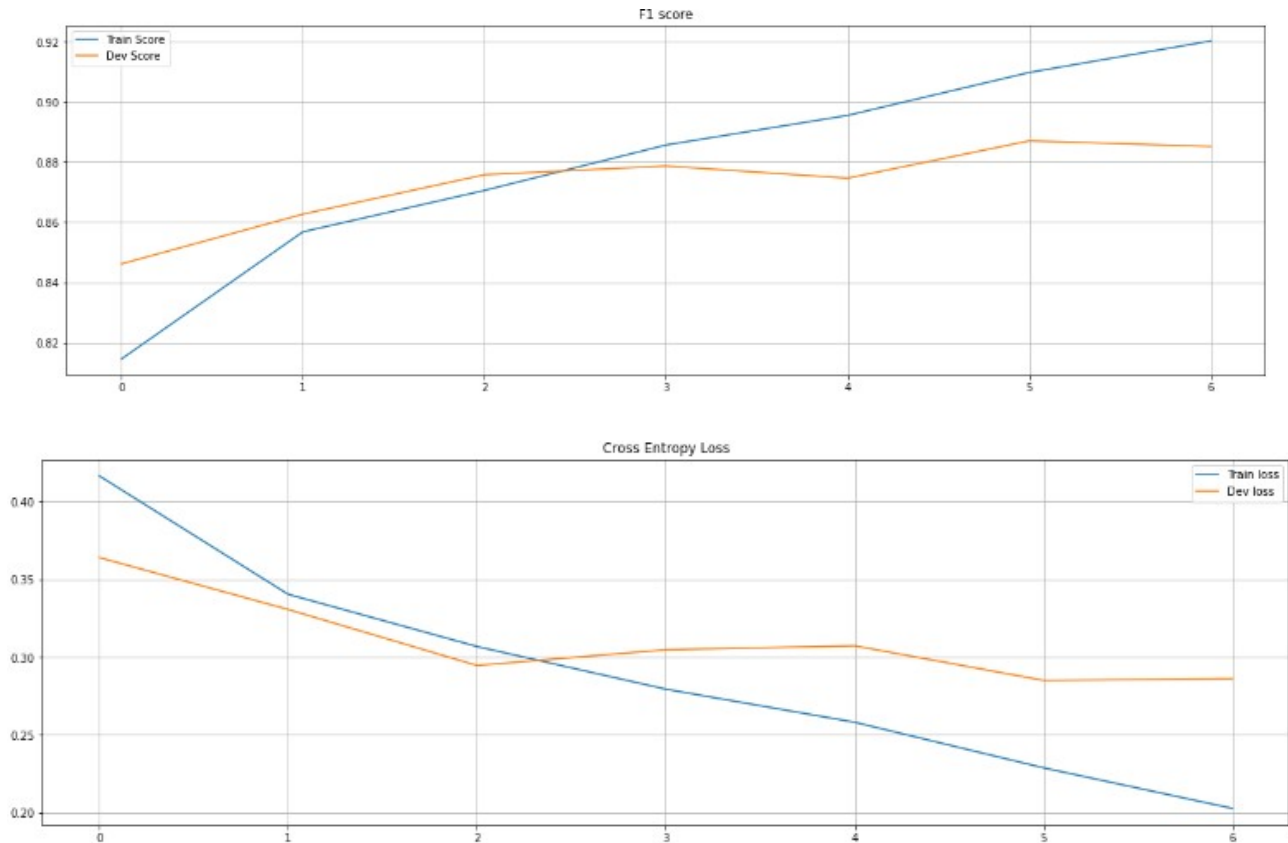
- At first, the length of reviews in the training set was analyzed so as to choose appropriate sequence length of the LSTM. Obtained distribution is shown below. It was found that most of the reviews had length of <100. If the sequence length was taken to be too high, lot of padding needs to be done which can be a source of confusion for the neural network while if it is taken to be too low, lot of tokens have to be truncated which would lead to losing many features from text. It was found that sequence length of 150 gave good performance.



- Initially a simpler LSTM (not BiLSTM) model was built without using pretrained word embeddings to check its performance. It was found to give F1 score of around 0.82 on the test set.
- The model was modified to BiLSTM and GloVe embeddings were added. Following loss and scores were obtained after each epoch.

```
Epoch 1 ...
train loss: 0.41661266156305105; dev loss : 0.3640461770387796
train score: 0.814522385597229; dev score : 0.8461538553237915
=====
Epoch 2 ...
train loss: 0.34041396399530083; dev loss : 0.33062945000636274
train score: 0.8568273186683655; dev score : 0.8627061247825623
=====
Epoch 3 ...
train loss: 0.30678420924605465; dev loss : 0.2945756174815007
train score: 0.8705923557281494; dev score : 0.8758038878440857
=====
Epoch 4 ...
train loss: 0.2793347770586992; dev loss : 0.3047055219992613
train score: 0.8856921195983887; dev score : 0.8787124156951904
=====
Epoch 5 ...
train loss: 0.2580505602348309; dev loss : 0.3072347171031512
train score: 0.8955350518226624; dev score : 0.8746576905250549
=====
Epoch 6 ...
train loss: 0.22870801718762288; dev loss : 0.2849727483131947
train score: 0.9097951054573059; dev score : 0.88707035779953
=====
Epoch 7 ...
train loss: 0.2026028199455677; dev loss : 0.2860434380097267
train score: 0.9202895164489746; dev score : 0.8852056264877319
=====
```

- Loss and scores of the model are plotted below with respect to the epoch number



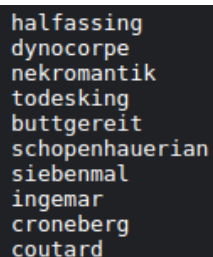
- Having done this, the F1 score on the test set went up to 0.88 which was a good improvement from the previous model.

Observations:

1. Since we have to pad or truncate each review to make their size equal to LSTM's sequence length, we could have either pre-padded & pre-truncated, or post-padded & post-truncated. It was found that pre-padding & pre-truncating yielded better results as compared to the other choice.
2. When the model was being trained, the F1 score on the training set kept increasing with every iteration and it even went beyond 0.99 when trained for 16 epochs. However, after some number of iterations, the loss on the validation set started increasing which was an indication that the model started overfitting. It was found that 7 epochs were sufficient to train the model, beyond which the loss on validation set started increasing and the F1 score started decreasing.
3. The formula for F1 score is $\frac{TP}{TP+0.5(FP+FN)}$ and if all the examples on which we compute F1 score are true negative, then the expression will be undefined. Hence, while splitting the data into train, dev and test, we must split in such a way that this case doesn't occur. Since, the dataset was fully balanced, i.e, 50% positive and 50% negative, we can always ensure that this doesn't happen.
4. One must ensure that while building the vocabulary, the tokens are either considered from some standard dictionary or obtained from training set only. In other words, the tokens from the reviews in the dev or test set must not be included. Otherwise we might get very good

results on this test set but when the model will be tested on some other dataset, the performance will be not that good.

5. The vocabulary size obtained from reviews in the training set after preprocessing the data was around 84k out of which around 20k tokens were not found in GloVe embeddings. The figure below shows some of the words which were not present in GloVe. Such tokens were treated as out of vocabulary tokens and were replaced with <UNK> tokens which were assigned some random embedding vector.



```
halfassing  
dynocorpe  
nekromantik  
todesking  
buttgerit  
schopenhauerian  
siebenmal  
ingemar  
croneberg  
coutard
```

Model:

The trained model can be downloaded from the link below:

<https://drive.google.com/file/d/1dMUZRnU-nun2EX1PqF4zHr3mzDDEv6KX/view?usp=sharing>