INLP Assignment 4 Report

Name - Pranav Gupta

Roll No. - 2021101095

As a general thumb of rule for the entire assignment, Sequence Length to train the LSTM for Downstream Classification Task was taken to be the length which was greater than 95 percent of the samples in the train set+1(so that pad tokens are not unnecessarily trained which would lead the model to bias towards the Pad tokens and thus hamper the training). This value came out to be 59. Glove Embeddings were taken as the Initial Pre-Trained Embeddings for the Assignmentin order to generate ELMo Embeddings. Since the dimension of each word in Glove Embedding is 100 and they need to be stacked on each other (we are taking BiLSTM Stack), so size fof each of the 3 embeddings (forward embeddings, backward embeddings and Glove embeddings) is 200. Firstly, ELMo class was trained in different modes to capture the Forward and Backward Embeddings while Glove Embeddings' role came in Downstream Classifiction Task. Learning Rate of 1e-3 and HIdden Dimension size = 200 were taken to train the LSTM (for downstream task). To train the LSTM, train set was divided in batches of 32. LSTM was trained for 10 epochs for both Trainable and Frozen Weights Cases of Hyperparameter Tuning.

2. Implementation and Training:

All the relevant code is written in 2 files according to the specifications of the Assignment (ELMO.py and classification.py). Jupyter Notebooks storing the graphs and results of each step are also attached for reference.

Performance of ELMo in Pretraining Procedure:

Train Dataset was split into Train and Validation (Dev) Datasets.

First of all, the Entire Train Dataset was divided into N-grams in order to perform Language Modeling with LSTM. A context window length of 5 was taken and using this context window we were predicting the next most probable word using LSTM analysis. A total of 3374491 N-grams were formed in this manner. However, because of computational issues, for the sake of simplicity, only the first 100000 ngrams were picked for training. Glove model was loaded. Then, corresponding to each word in the Train Dataset, embedding matrix was created which was given as input to the ELMo model for creation of forward and backward embeddings.

Then the forward embeddings were trained by optimising it with the forward LSTM and the same was done with the Backward Embeddings. Each of the 2 embeddings were trained for 4 epochs and the following loss values were reported:

Forward Embeddings:

Epoch = 1 → Train Loss: 0.0469
Epoch = 2 → Train Loss: 0.0371
Epoch = 3 → Train Loss: 0.0323
Epoch = 4 → Train Loss: 0.0290

Backward Embeddings:

Epoch = 1 → Train Loss: 0.0468
Epoch = 2 → Train Loss: 0.0370
Epoch = 3 → Train Loss: 0.0322
Epoch = 4 → Train Loss: 0.0288

Then the Pre-Trained model was saved to be used later for Downstream Classification Task.

4. Hyperparameter Tuning:

Hyperparameters used to train the model were taken the same as specified above. Only, the weights and Gamma value was varied in this procedure and the Evaluation Metrics were noted.

As specified in the Assignment PDF, 2 cases were considered for Training. (1) When the Weights were Trainable, and (2) when the weights were Frozen.

1. Trainable Weights Case:

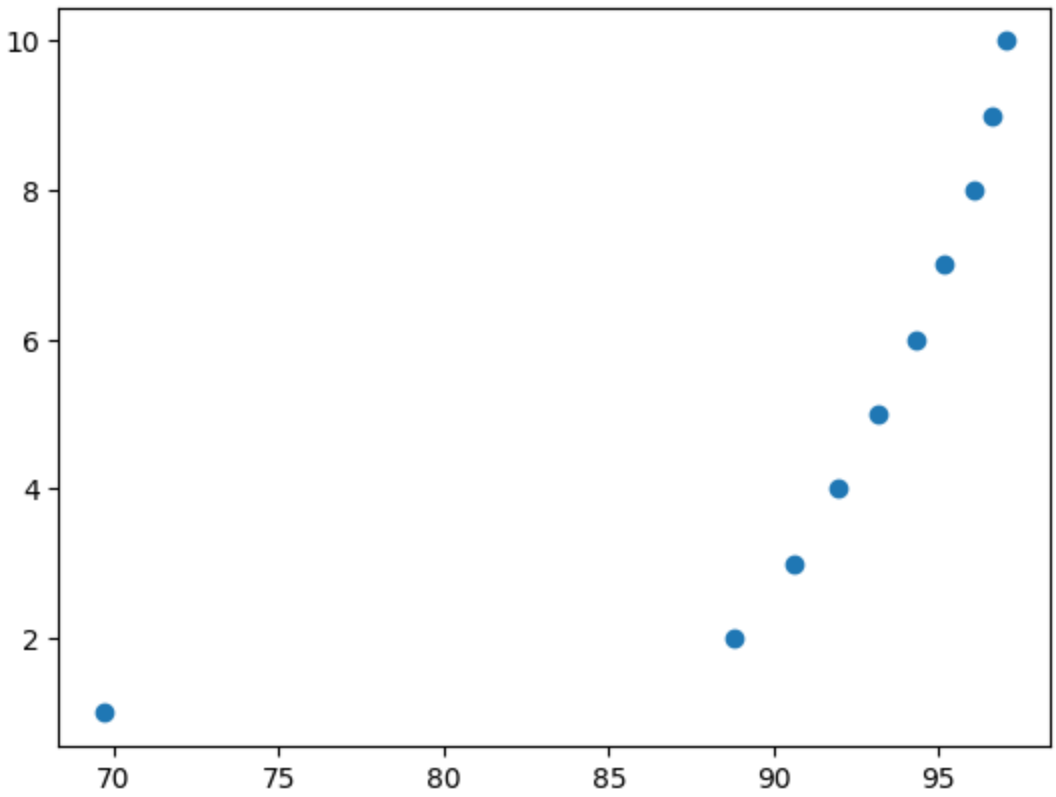
   Initially Lambdas were taken uniformly across the 3 classes (0.33, 0.33 and 0.33) while the Scaling Factor Gamma was taken as 1. After the Training Process, lambda values were found out to be 0.2493, 0.3161, 0.4346 (using Normalized Softmax Operations) while gamma was evaluated as 1.6485.

   Evaluation Metrics Obtained on Train Set after 10 epochs in this case are as follows:

   Train Set Accuracy = 97.024
   Train Set Recall = 97.024
   Train Set F1-Score = 94.83753983076268
   Train Set Precision = 94.93347362900482
   Confusion Matrix:
   [[23281 50 284 272]
   [481 24875 152 131]
   [743 42 22599 582]
   [527 53 1854 24074]]

   Evaluation Metrics Obtained on Test Set in this case are as follows:

   Test Set Accuracy = 89.78
   Test Set Recall = 89.78
   Test Set F1-Score = 89.79872608594803
   Test Set Precision = 89.94152474201607
   Confusion Matrix:
   [[1124 10 30 35]
   [50 1243 12 13]
   [64 9 1011 81]
   [48 8 151 1111]]



Training Set Accuracy Vs Number of Epochs in Downstream Classification Task

2. Frozen Weights Case:

   Initially Lambdas were taken as random in this case, and the Values of Lambdas (0.7442, 0.1868, 0.0690) and Gamma = 1 remained the same even after the Training Process because this time the Weights and Value of Gamma was frozen.

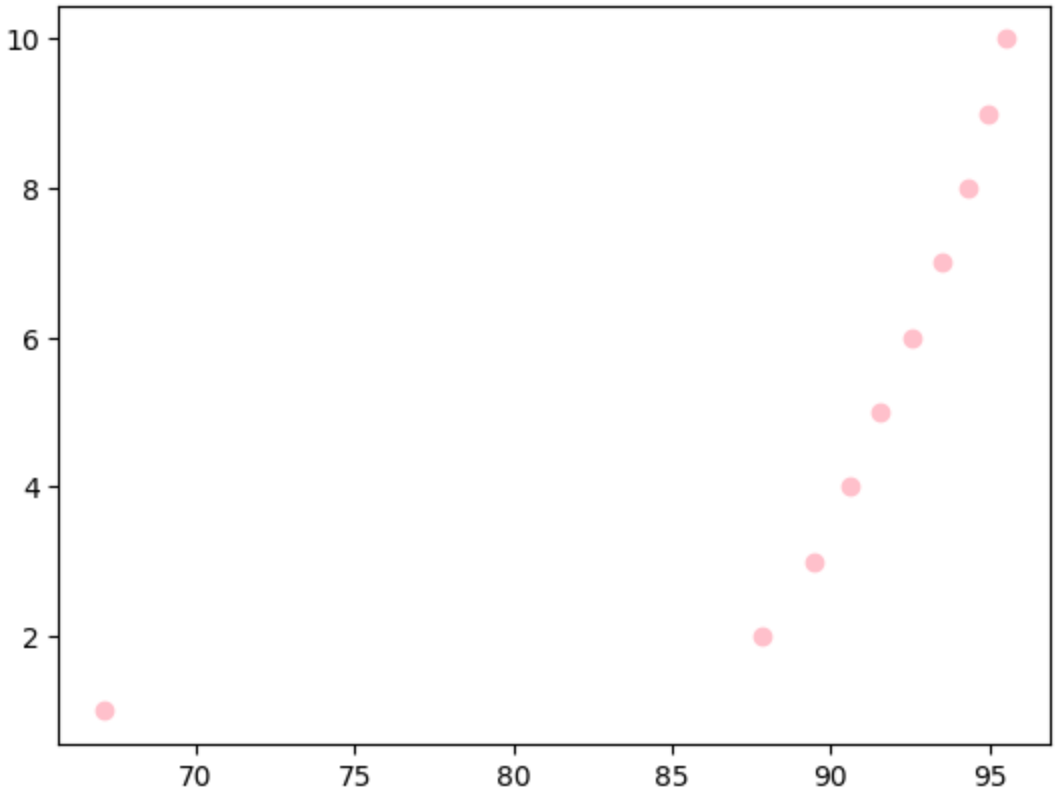
   Evaluation Metrics Obtained on Train Set after 10 epochs in this case are as follows:

   Train Set Accuracy = 95.507
   Train Set Recall = 95.507
   Train Set F1-Score = 93.42824243025008
   Train Set Precision = 93.54659342881327
   Confusion Matrix:
   [[23447 177 506 488]

[352 24630 162 146]
[625 91 21730 812]
[608 122 2491 23613]]

Evaluation Metrics Obtained on Test Set in this case are as follows:

Test Set Accuracy = 88.78
Test Set Recall = 88.78
Test Set F1-Score = 88.80254126888259
Test Set Precision = 88.9050533896645
Confusion Matrix:
[[1131 18 43 43]
[47 1229 15 18]
[57 9 992 92]
[51 14 154 1087]]



Training Set Accuracy Vs Number of Epochs in Downstream Classification Task

5. Analysis:

The corresponding Evaluation Metrics obtained for SVD model are as follows:

1. Train Set:

   Training Set Accuracy = 82.25166666666667

   Recall = 82.25166666666667

   Precision = 83.21663927915915

   F1-Score = 82.23682537651327

2. Test Set:

   Testing Set Accuracy = 72.75

   Recall = 72.75

   Precision = 73.1093351800554

   F1-Score = 72.88439959010665

The corresponding Evaluation Metrics obtained for Skip-Gram model are as follows:

1. Train Set:

   Training Set Accuracy = 93.00333333333334

   Recall = 93.00333333333333

   Precision = 93.6421206606781

   F1-Score = 93.01234429207267

2. Test Set:

   Testing Set Accuracy = 85.68421052631578

   Recall = 85.68421052631578

   Precision = 86.38512465373962

   F1-Score = 85.81816337286763

We observe that out of the 3 configurations given above (ELMo, Skip-Gram and SVD), ELMo slightly outperforms Skip-Gram Embeddings while on the other hand perform much better than SVD word Embeddiings. We note that almost all the metrics for ELMO are better compared to that of Skip-Gram and SVD.

Comparison of the 3 Embeddings:

ELMo captures the meaning of a word based on its surrounding context, leading to more nuanced representations compared to SVD, which focus on word co-occurrence or document statistics. This context-sensitivity allows ELMo to handle word ambiguity and polysemy better. Skip-gram also does this and because of this reason it is much better than SVD while does not perform as good as ELMo because it does take into account the Backward dependencies or the Bidirectional Learning Component of ELMo. Since, ELMo uses both left and right contexts, it is able to capture better the dependencies better than Skip-gram. Also, it uses Stacked Layers, which allows it learn complex relationships between words and learn higher-order semantic features.

SVD focuses on dimensionality reduction, which while useful for capturing overall document structure, might discard valuable semantic information in the process. It takes each of the word as a separate entity and then creates embeddings and completely ignores the neighbouring context while on the other hand, skip-gram and ELMO , because of the consideration of the context, are able to learn Word Vector representations very efficiently. This is one of the important reasons why ELMO performs better than Skip-gram and skip-gram performs better than SVD in general.

In case of Skip-Gram, Embedding Dimension of 300 was considered along with LSTM Hidden Dimension of 128. In case of SVD, Embedding Dimension of 300 was considered along with LSTM Hidden Dimension of 128 while for ELMO Embedding Dimension of 200 was considered along with Hidden Dimension of LSTM as 200. Also, one thing to note here is that Skip-Gram and SVD were trained for Dimension of 300 while ELMO is trained for 200 dimensions only. Secondly, in case of generation of forward and backward

LSTMs, code was run for only 4 epochs and if run for larger no. of epochs, the pre-trained word embeddings would be far better. Also, last but not the least, only the first 100000 ngrams were taken for training of Forward and Backward Embeddings out of 3374491 ngrams. If trained on entire dataset, contextual embeddings will be better. In short, if ELMO still performs better even when there were so many limitations compared to Skip-gram and SVD, that means ELMO is a powerful algorithm for generating Deep Contextualized Embeddings.