ASSIGNMENT – 4

Sundeep Rachuri

PURPOSE:

The goal of this job is to apply RNNs or transformers to both text and sequence data. I'm focussing on the IMDB movie review dataset in order to predict sentiment in reviews. I have restricted the entire dataset to the top 10,000 phrases out of 50,000 reviews for the purpose of simplicity. In order to see how it affects the model's performance, I plan to change the number of reviews in my small training sample of 100. My main objective is to compare the performance of multiple models with different embedding layers and training sample sizes. I'm curious to see how these changes impact the algorithms' accuracy in classifying reviews of films as either positive or negative.

I intend to learn more about the trade-offs between prediction accuracy, training data size, and model complexity by experimenting with these factors. This should make it clearer to me when using a more complex model with bigger datasets may be preferable than using a simpler model with less data.

APPROACH:

I used two different approaches to the task of making word embeddings for the dataset. In the first approach, a bespoke embedding layer was created, and in the second, an existing word embedding based on the GloVe model was used. I created two distinct configurations to test the efficacy of these methods: one that used the pre-trained word embedding layer and the other that used the custom-trained embedding layer I had created. I was able to evaluate the effectiveness of both strategies using this configuration, which gave me insight into their advantages and possible uses in the model.

To assess the accuracy of the two models, I experimented with training sample sizes of 100, 500, 1000, and 10,000. I started by using the IMDB review dataset to develop a custom-trained embedding layer. I used a testing set to gauge each model’s accuracy after training it on various dataset samples. I then tested a model with a pre-trained word embedding layer on the same sample sizes and compared the results. Based on the data, I discovered that, depending on the training sample size, the accuracy of the embedding layer learnt from scratch varied between 78.4% and 81.1%. The accuracy of the pre-trained word embedding layer ranged from 78.2% to 80.7%. Notably, the model with high test accuracy that was trained on 10,000 samples did exceptionally well.

By using this method, I was able to assess the performance of the pre-trained and custom-trained embedding layers on a range of dataset sizes, providing me with information about their respective accuracies and efficiency.

RESULTS:

MODEL	TRAINING SAMPLES SIZE	TRAINING LOSS	VALID LOSS	TRAINING ACCURACY	VALIDATION ACCURACY	TEST ACCURACY
Basic Sequence Model	100	0.85	57.5	97.5	77.8	80.6
Embedding layer from scratch	100	0.24	72.1	99.4	76.2	78.4
Embedding layer from scratch	500	0.13	84.8	99.6	81.1	80.6
Embedding layer from scratch	1000	0.08	94.2	99.8	81.2	80.6
Embedding layer from scratch	10000	0.08	1.06	99.7	79.9	81.1
Pre-trained word embedding	100	43.7	49.4	80.6	76.3	78.2
Pre-trained word embedding	500	0.08	1.12	99.6	80.8	79.4
Pre-trained word embedding	1000	0.12	88.3	99.7	80.8	79.5
Pre-trained word embedding	10000	0.11	97.2	99.7	79.7	80.7

Based on the data, I discovered that, depending on the training sample size, the accuracy of the embedding layer learnt from scratch varied between 78.4% and 81.1%. The accuracy of the pre-trained word embedding layer ranged from 78.2% to 80.7%. Notably, the model with high test accuracy that was trained on 10,000 samples did exceptionally well.

Given these results, it is challenging to say with certainty which approach is “best” because it primarily depends on the unique needs and limitations of every project. However, I discovered that the embedding layer that was learnt from scratch outperformed the pre-trained word embedding in this study, particularly with larger sample numbers.

I would incline towards suggesting the embedding layer built from scratch if there are restricted computer resources and only a small training sample size is available. However, it's crucial to use caution when overfitting. The 10,000-sample model could be a fair compromise when time and processing resources are constrained. In the end, the choice between these approaches would depend on factors like the available resources, the dataset's complexity, and the specific requirements of the task at hand. It finds a balance between usability, efficacy, and functionality.