ASSIGNMENT – 4
Goli Sai Priyanka

PURPOSE:

Applying RNNs or transformers to text as well as sequence data is the aim of this task. To forecast sentiment in reviews, I'm concentrating on the IMDB movie review dataset. For simplicity's sake, I have limited the complete dataset to the top 10,000 terms out of 50,000 reviews. I intend to modify the quantity of reviews in my tiny training sample of 100 to observe how it impacts the model's performance. Comparing the performance of several models with varying training sample sizes and embedding layers is my major goal. I'm interested to observe how these modifications affect the models' precision in categorizing movie reviews as favorable or unfavorable.

By experimenting with these variables, I hope to gain insights into the trade-offs between model complexity, training data size, and prediction accuracy. This should help me understand when it might be better to use a simpler model with less data versus a more complex one with larger datasets.

APPROACH:

I approached the task of creating word embeddings for the dataset using two distinct methods. The first method involved developing a custom embedding layer, while the second utilized a pre-existing word embedding based on the GloVe model. To evaluate the effectiveness of these approaches, I set up two separate configurations: one incorporating the custom-trained embedding layer I had developed, and another using the pre-trained word embedding layer. This setup allowed me to compare and contrast the performance of both approaches, providing insight into their respective strengths and potential applications in the model.

I experimented with a variety of training sample sizes (100, 500, 1000, and 10,000) to evaluate the accuracy of the two models. First, I created a custom-trained embedding layer using the IMDB review dataset. After training each model on different dataset samples, I measured its accuracy using a testing set. Then, I compared these results with a model that had a pre-trained word embedding layer, testing it on the same sample sizes. Based on the data, I found that the accuracy of the embedding layer trained from scratch ranged from 78.4% to 81.1%, depending on the training sample size. The pre-trained word embedding layer achieved accuracies between 78.2% and 80.7%. Notably, the model trained on 10,000 samples performed particularly well, showing high test accuracy.

This approach allowed me to evaluate how the custom-trained and pre-trained embedding layers performed across various dataset sizes, giving me insights into their relative efficiencies and accuracies.

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 found that the accuracy of the embedding layer trained from scratch ranged from 78.4% to 81.1%, depending on the training sample size. The pre-trained word embedding layer achieved accuracies between 78.2% and 80.7%. Notably, the model trained on 10,000 samples performed particularly well, showing high test accuracy.

Considering these findings, it is difficult to declare with certainty which strategy is "best" because it largely depends on the particular requirements and constraints of each project. Nevertheless, I found that in this research, especially with higher sample sizes, the embedding layer that was learned from scratch performed better overall than the pre-trained word embedding.

If computing resources are limited and only a modest training sample size is available, I'd lean towards recommending the embedding layer trained from scratch. It's important to be cautious about overfitting, though. When faced with limited time and processing resources, the 10,000-sample model might be a good middle ground. The decision between these methods would ultimately come down to things like the resources that are available, the complexity of the dataset, along with the particular needs of the work at hand. It strikes a balance between functionality, effectiveness, and ease of use.