Assignment-4

Apply RNNs to text and sequence data

Department of Information Systems and Business Analytics,

Kent State University

Spring 2025 ADVANCED MACHINE LEARNING (BA-64061-001)

Name: Jayanth Yadlapalli
Student Id: 811189968

1. Purpose

This task aims to investigate the application of Recurrent Neural Networks (RNNs) in text and sequence data analysis. The task has the following objectives:

1.Apply RNNs to IMDB sentiment analysis data.

2. Demonstrate procedures to improve model performance, especially in handling small data.

3. Compare and establish the effectiveness of two approaches:

Application of a custom embedding layer.

Application of pre-trained word vectors (GloVe).

2. Dataset Overview

The IMDB dataset, containing labeled movie reviews, was used for this analysis. Key preprocessing steps included:

1. Cutting off reviews after 150 words.
2. Restricting training samples to 100**.**
3. Validating on 10,000 samples.
4. Considering only the top 10,000 words in the vocabulary.

3. Model Architectures

Two models were trained for this task:

1.Custom Embedding Model:

Embedding layer with 128 dimensions.

A bidirectional LSTM layer with 32 units.

Dropout for regularization (rate = 0.5).

Dense output layer with a sigmoid activation function.

Layer	**Output Shape**	**Param #**
Input	(None, None)	0
Embedding	(None, None, 128)	1,280,000
Bidirectional LSTM	(None, 64)	41,216
Dropout	(None, 64)	0
Dense (Sigmoid)	(None, 1)	65
Total Parameters		**1,321,281**

Pretrained Embedding Model (GloVe):

Pretrained GloVe embeddings (100 dimensions).

Embedding layer initialized with pretrained weights (non-trainable).

A bidirectional LSTM layer with 32 units.

Dropout for regularization (rate = 0.5).

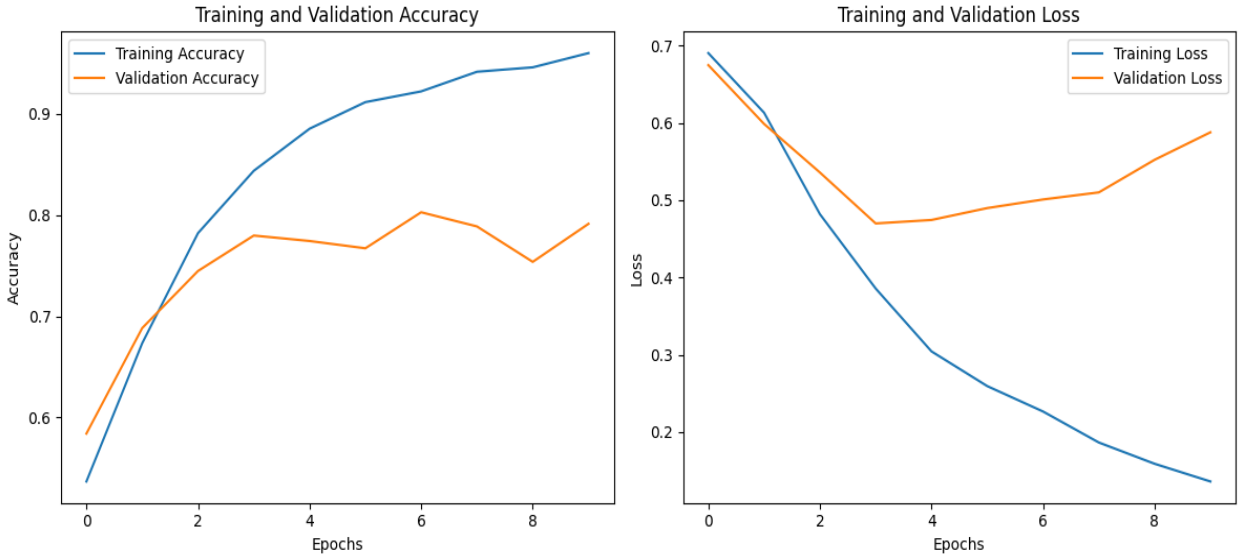
Dense output layer with a sigmoid activation function.

Layer	**Output Shape**	**Param #**
Input	(None, None)	0
Embedding (GloVe)	(None, None, 100)	1,000,000
Bidirectional LSTM	(None, 64)	34,048
Dropout	(None, 64)	0
Dense (Sigmoid)	(None, 1)	65
Total Parameters		**1,034,113**

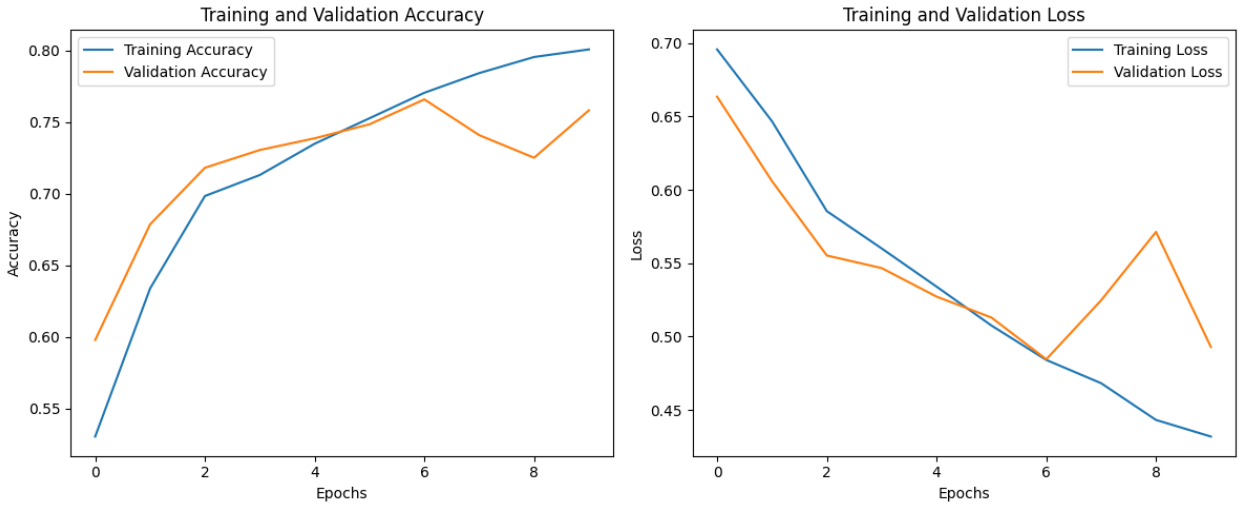
4. Results and Evaluation

a. Training and Validation Accuracy and Loss Plots for both models are presented below, showing trends in training and validation accuracy and loss over 10 epochs.

Custom Embedding Model Accuracy and Loss

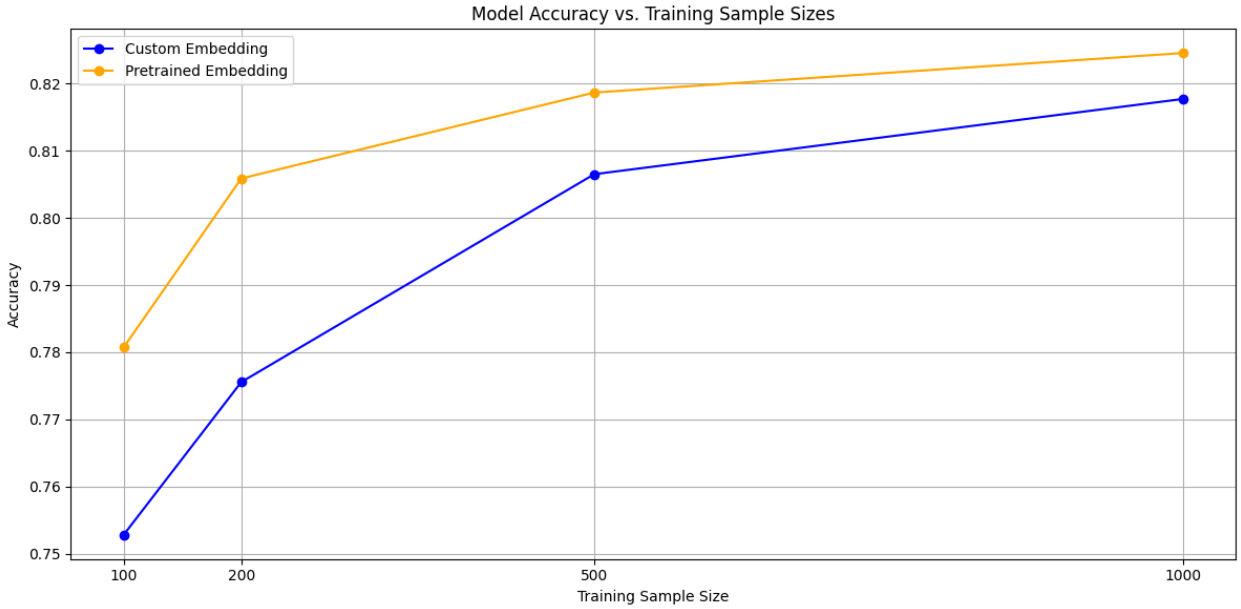


Pretrained Embedding Model Accuracy and Loss



Effect of Sample Size on Model Accuracy To determine how sample size impacts performance, both models were trained on varying sample sizes (100, 200, 500, and 1,000).

Comparison Plot: Final Test Accuracy vs. Sample Size



Sample Size	Custom Embedding Accuracy	Pretrained Embedding Accuracy	Difference
100	0.7528	0.7807	0.0279
200	0.7756	0.8059	0.0303
500	0.8065	0.8187	0.0122
1,000	0.8177	0.8246	0.0069

Analysis of Results

The summary table of accuracies for custom embedding and pretrained embedding models at different training sample sizes highlights distinct trends and provides insights into the performance of the two approaches under varying data constraints.

2. Observations

1.Accuracy Gap Between Custom and Pretrained Models:

Small Sample Sizes (100–200):

The pretrained embeddings significantly outperformed the custom embeddings with differences of 2.79% at 100 samples and 3.03% at 200 samples.

This highlights the ability of pretrained embeddings to leverage prior linguistic knowledge when data is scarce.

Larger Sample Sizes (500–1,000):

The gap narrows as the sample size increases, with differences reducing to 1.22% at 500 samples and 0.69% at 1,000 samples.

Custom embeddings start to perform more competitively as they benefit from task-specific feature learning with increasing data.

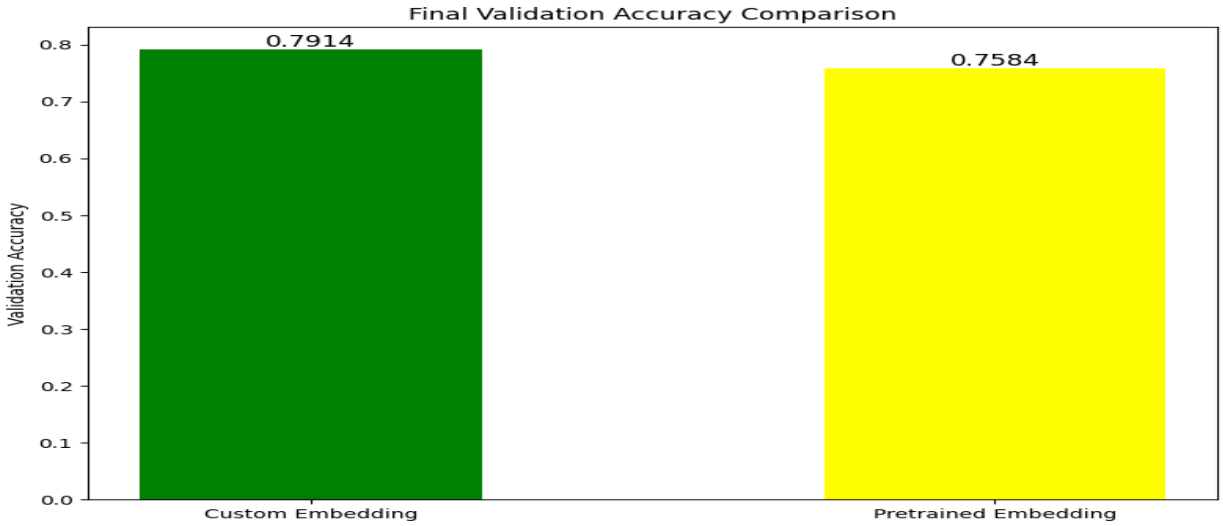
Improvement in Accuracy with Increased Sample Size:

Both models show consistent improvements in accuracy as the sample size increases.

Custom Embeddings improved from **0.7528 (100 samples)** to **0.8177 (1,000 samples)** an absolute increase of 6.49%.

Pretrained Embeddings improved from **0.7807 (100 samples)** to **0.8246 (1,000 samples)** an absolute increase of 4.39%.

The improvement trend is more pronounced for the custom embeddings, indicating that these models require more data to generalize effectively.



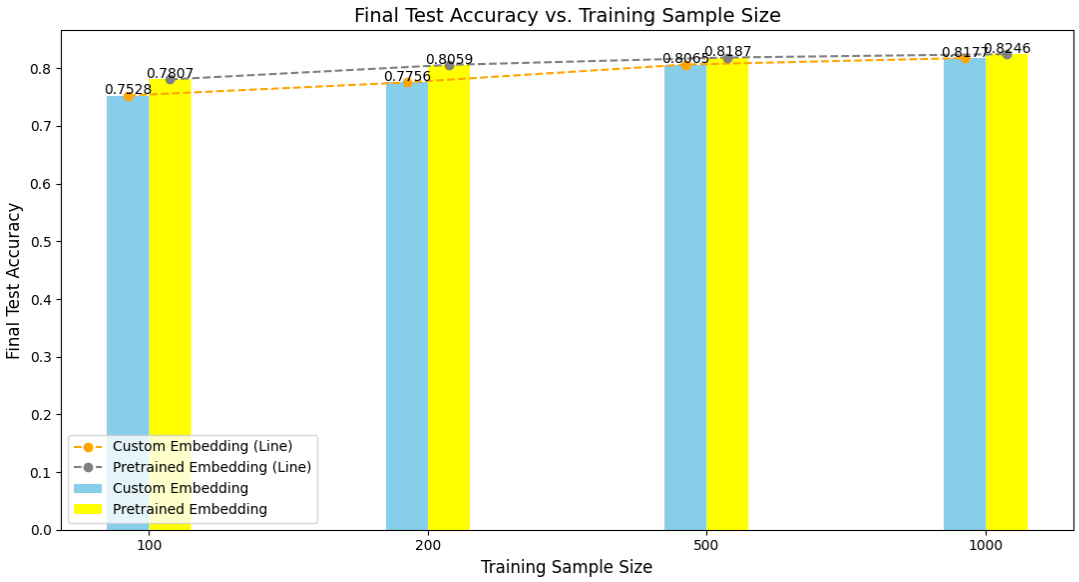
Plateauing Effect:

Accuracy improvement for methods gets saturated while approaching a sample size of 1,000.

For instance, the improvement in accuracy from 500 to 1,000 samples is not as great as with past increases in sample size:

Custom: 0.8065 -0.8177 (+1.12%)

Pretrained: 0.8187 - 0.8246 (+0.59%)



This indicates diminishing returns on more data and points to the need for other optimizations like hyperparameter tuning or changes to the model structure.

3. Key Takeaways

1. Pretrained Embedding Performance:

Pretrained embeddings perform better in low-data conditions on the basis of their prior semantic and syntactic relation knowledge acquired at pretraining from large corpora.

They are good generalizers, even for small task-specific data, with better performance.

2. Dependence of Custom Embeddings on Data:

There is a need for more task-specific data to learn useful representations by custom embeddings, which reduces their performance on smaller sample sizes.

But with sufficient data, they can match performance, even surpassing pretrained embeddings in some cases.

3. Transition Point:

The performance gap between the two approaches reduces at 500–1,000 samples. This is a transition point where task-specific learning with custom embeddings starts to bridge the gap on general knowledge in pretrained embeddings.

4. Scalability:

Pretrained embeddings are applicable for projects with limited data availability.

Custom embeddings are better suited for larger data and domain-specific task requirements.

4. Strategic Recommendations

1. Small Data Scenarios (<500 samples):

Use pretrained embeddings to ensure better generalization and performance.

Fine-tune the embeddings if possible to adapt them to the task-specific domain.

2. Moderate to Large Data Scenarios (>1,000 samples):

Custom embeddings can be employed as they can match or outperform pretrained embeddings with sufficient data.

Experiment with other model augmentations such as:

Dim increasing.

Increasing LSTM units or additional layers.

Utilizing attention mechanisms to capture more context.

3. Hybrid Approaches:

Try hybrid approaches like pre-training embeddings with pretrained weights and making it trainable during task-specific training. This is a mixture of the two.

5. Conclusion

The comparison highlights the clear advantage of pretrained embeddings under low-data scenarios and their strengths of generalizability. When additional data become accessible, task-specific complexities, however, can be leveraged by tailored embeddings to match or even surpass. The methodology choice is a matter of dataset size, domain usage, and computational cost.