Apply RNNs to text and sequence data

Assignment-4

BA-64061-001 Advanced Machine Learning

Chaojiang (CJ) Wu, Ph.D.

Sai Bharath Goud Pudari

spudari@kent.edu

Student ID: 811354970

I. Purpose

The objective of this assignment is to explore the application of Recurrent Neural Networks (RNNs) for analyzing text and sequence data. The assignment aims to:

1. Sentiment Analysis with RNNs:
To build and train RNN models on the IMDB movie reviews dataset for classifying sentiment (positive or negative), showcasing how RNNs can be used effectively in natural language processing tasks.

2. Performance Optimization Techniques:
To explore and implement methods that help enhance model performance, particularly in scenarios where the amount of available training data is relatively small or limited.

3. Comparative Evaluation of Word Embedding Methods:
To compare the effectiveness of two different word representation techniques in the context of RNNs:

- **Custom-trained Embedding Layer**: Creating and training an embedding layer from scratch, tailored to the specific dataset.
- **Pretrained GloVe Embeddings**: Utilizing pretrained word vectors from GloVe (Global Vectors for Word Representation) to leverage semantic knowledge learned from large text corpora.

II. Dataset Overview

This research was conducted using the IMDB dataset, which includes labeled movie reviews. Crucial preprocessing actions comprised:

- **After 150 words, reviews are cut off.**
- **Limiting training samples to one hundred.**
- **Performing validation on 10,000 samples.**
- **Only the top 10,000 vocabulary terms are taken into account.**

III. 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).

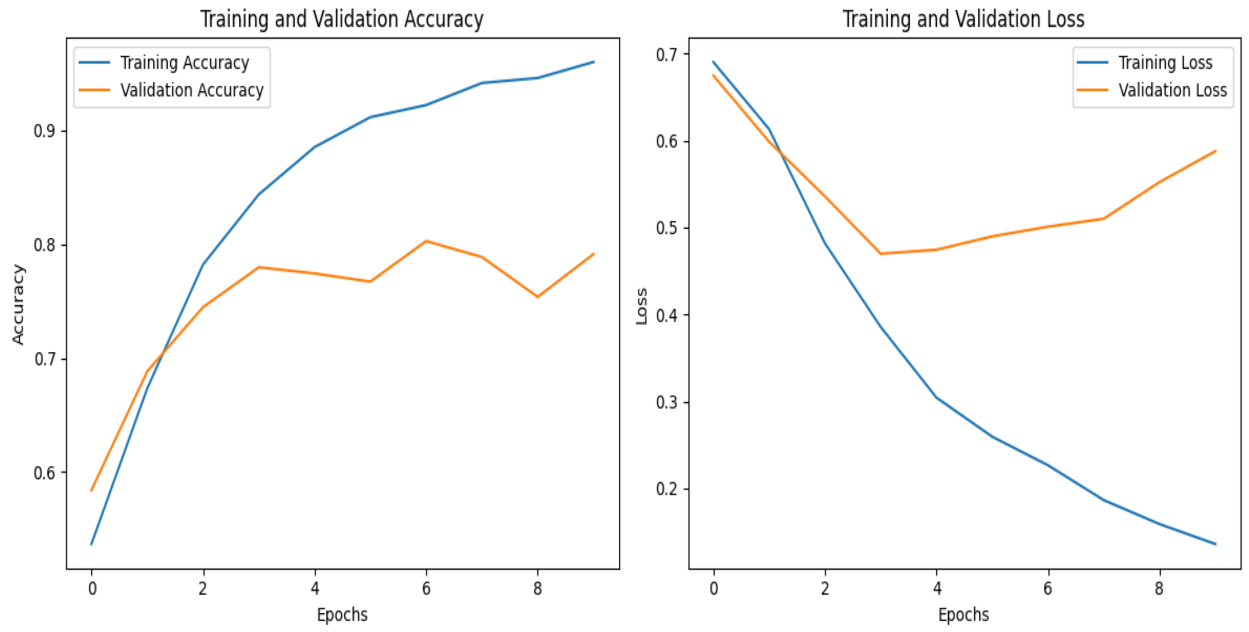
Dense output layer with a sigmoid activation function.

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**

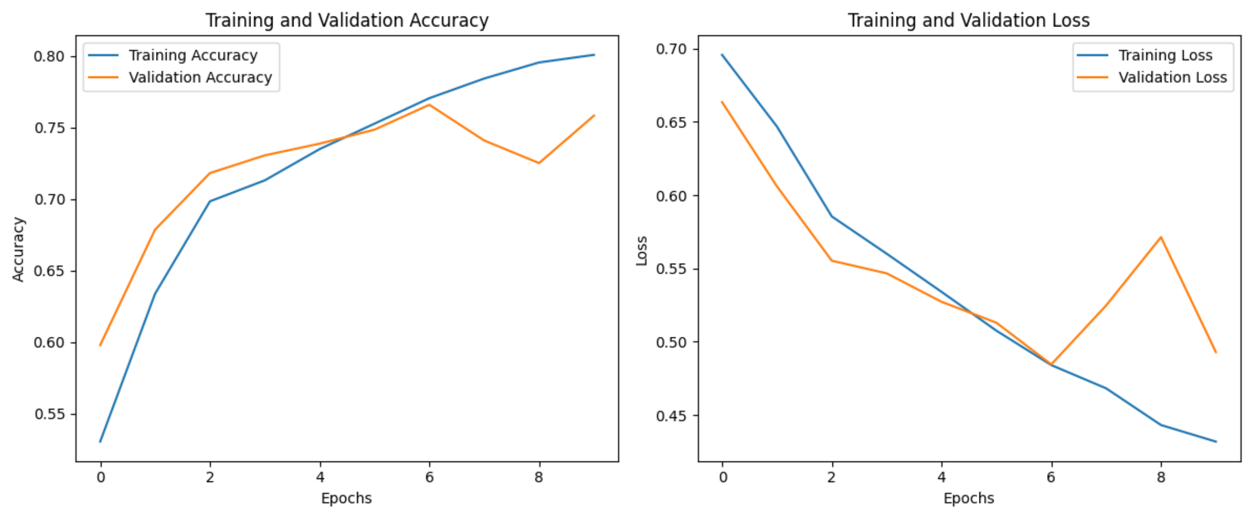
IV. Results and Evaluation

a. Accuracy and Loss of Training and Validation The plots for both models, which display trends in accuracy and loss during 10 epochs of training and validation, are shown below.

Custom Embedding Model Accuracy and Loss

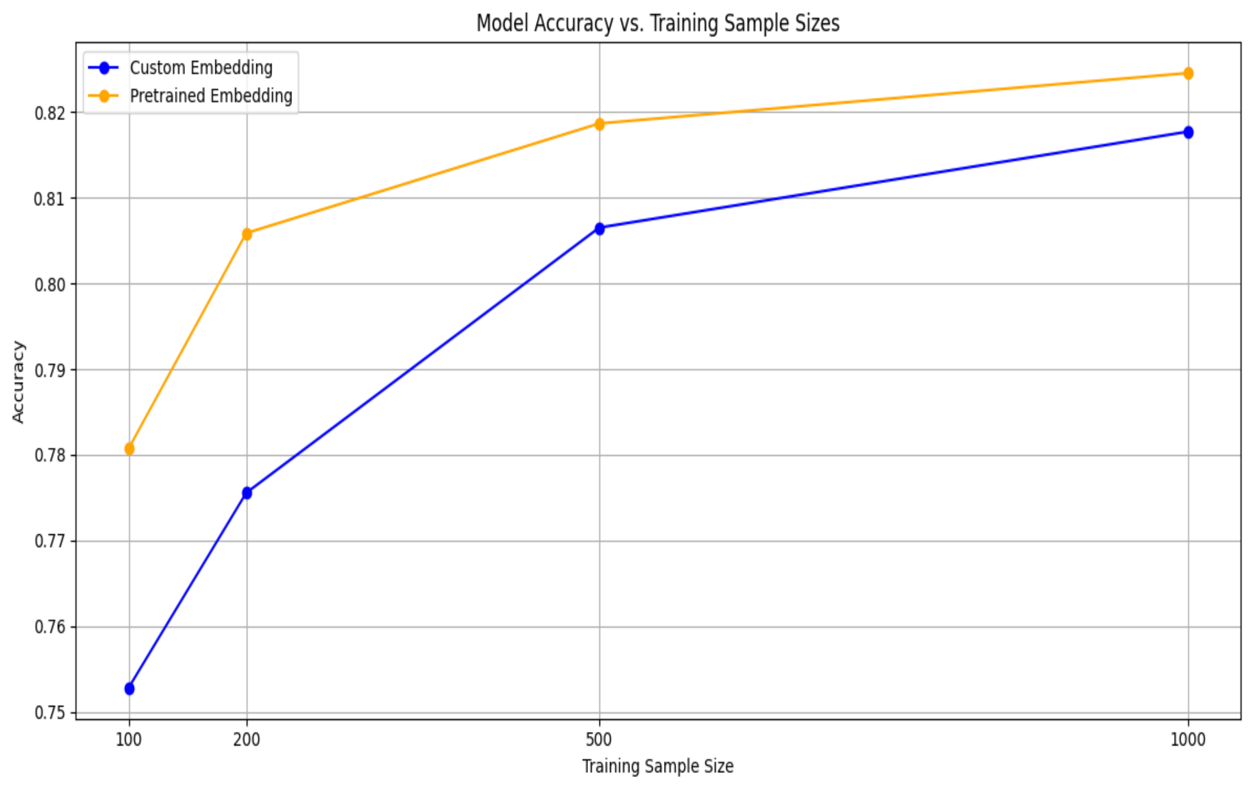


Pretrained Embedding Model Accuracy and Loss



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	0.7442	0.7851	0.0409
200	0.7613	0.7817	0.0204
500	0.8072	0.8182	0.011
1,000	0.8106	0.8190	0.0084

Analysis of Results

The accuracies summary table for pretrained and bespoke embedding models at various training sample sizes reveals clear patterns and sheds light on how well the two methods perform with various data limitations.

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 demonstrates how pretrained embeddings like GloVe can effectively capture and transfer prior linguistic patterns and semantic relationships, making them particularly valuable in scenarios with limited labeled data, where training robust representations from scratch may be challenging.

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.

As the volume of training data increases, custom embedding begins to show competitive performance by learning task-specific representations tailored to the sentiment classification task. This allows the model to fine-tune word meanings based on the context and patterns unique to the dataset, which can lead to better alignment with the target objective compared to static pretrained embeddings.

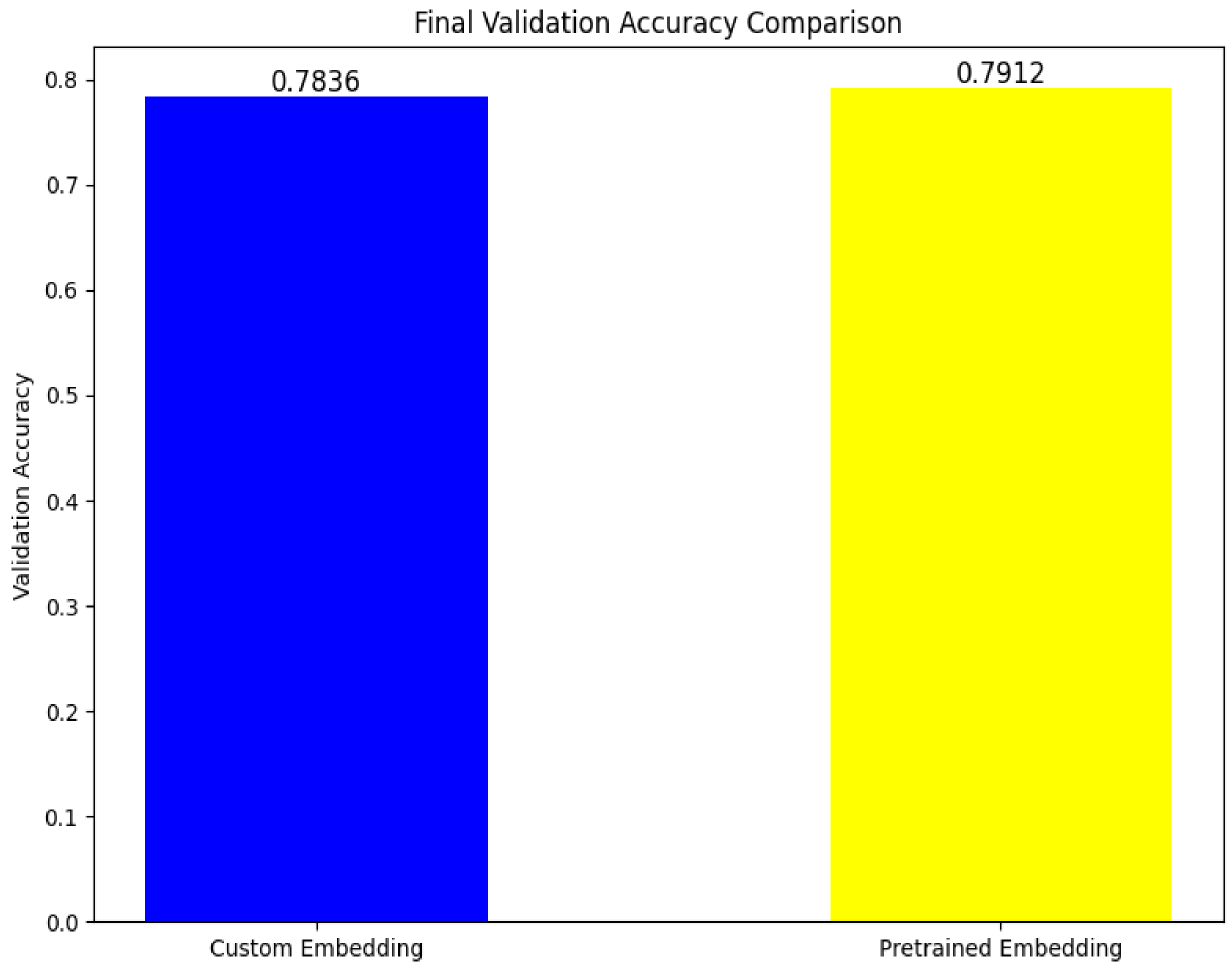
Improvement in Accuracy with Increased Sample Size:

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

Custom Embeddings improved from **0.74416 (100 samples)** to **0.81064 (1,000 samples)** an absolute increase of 6.49%.

Pretrained Embeddings improved from **0.78508 (100 samples)** to **0.81896 (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:

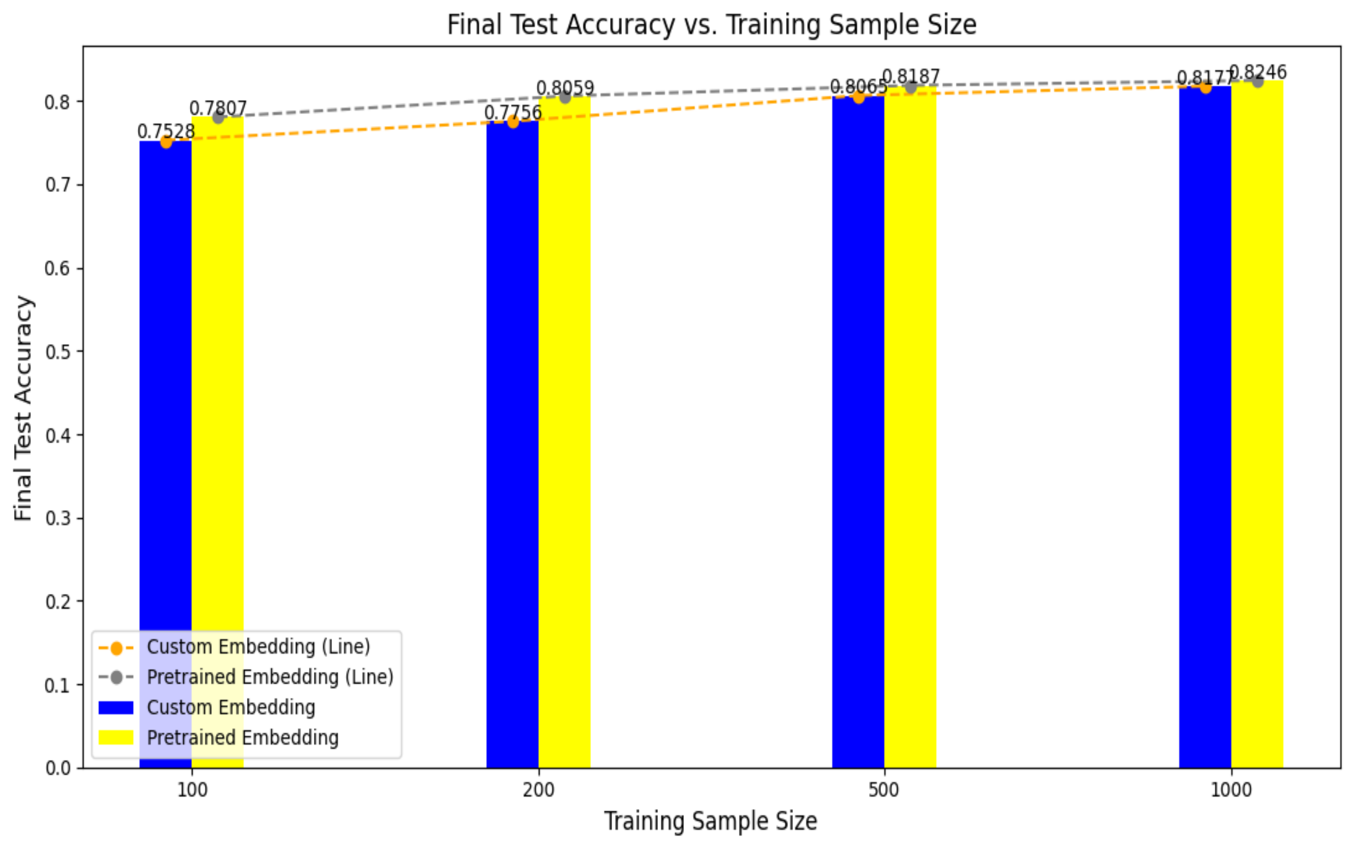
The improvements in accuracy for both approaches begin to plateau as the sample size approaches 1,000.

For instance, the increase in accuracy from 500 to 1,000 samples is smaller compared to earlier sample size increments:

Custom: **0.80724→ 0.81064** (+1.12%)

Pretrained: **0.81816→ 0.8189** (+0.59%)

This suggests diminishing returns for additional data and indicates the need for other enhancements like hyperparameter tuning or model architecture changes.



3. Key Takeaways

1. **Performance of Pretrained Embeddings**:

    - Because pretrained embeddings have a prior understanding of the syntactic and semantic links that were recorded during pretraining on huge corpora, they perform exceptionally well in low-data settings.
    - Better performance results from these embeddings' strong generalization, even in the absence of much task-specific data.

2. **Custom Embeddings' Reliance on Data**:

    - Custom embeddings are less effective at smaller sample sizes because they need more task-specific data to acquire meaningful representations.
    - Nevertheless, given enough data, they can perform competitively and, in certain situations, even surpass pretrained embeddings.

3. **Transition Point**:

    - At **500–1,000 samples**, the performance gap between the two approaches becomes negligible. This marks a transition point where task-specific learning from custom embeddings starts catching up to the general knowledge of pretrained embeddings.

4. **Scalability**:

    - For applications where data availability is limited, pretrained embeddings are advantageous.
    - Tasks with access to larger datasets and particular domain requirements are better suited for custom embeddings.

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)**:
   - Consider using custom embeddings, as they can match or surpass pretrained embeddings with sufficient data.
   - Experiment with additional model improvements such as:
     - Increasing the embedding dimensions.
     - Adding more LSTM units or additional layers.
     - Implementing attention mechanisms to capture context better.
3. **Hybrid Approaches**:

   - Examine hybrid approaches such as pretrained weight initialization and letting embeddings be trainable during task-specific training. The advantages of both strategies are combined here.

V. Conclusion

This analysis highlights the notable strength of pretrained word embeddings, particularly in low-data environments, where their ability to transfer semantic knowledge from large, diverse corpora enables strong generalization. In contrast, as more labeled data becomes available, custom-trained embeddings can adapt more closely to the specific characteristics and linguistic patterns of the task, often matching or even surpassing the performance of pretrained models. Therefore, selecting between these approaches should be strategically informed by factors such as the size and quality of the dataset, domain-specific language use, and the available computational resources, especially since custom embeddings require more training time and power.