BA-64061 Advanced Machine Learning

Assignment-4

Apply RNNs to text and sequence data

Name: Chandra Lekha Suggala

Student ID: 811344864

E-Mail ID: csuggala@kent.edu

Assignment 4 Summary Report: Apply RNNs To Text And Sequence Data

Objective:

The goal of this project is to do sentiment analysis on the IMDB movie reviews dataset using Recurrent Neural Networks (RNNs), more especially LSTM models. Preprocessing text data, restricting evaluations to 150 words, and utilizing just the 10,000 most often occurring terms are all part of the work. Pretrained GloVe word embeddings and a custom trainable embedding layer are used in the two modeling techniques that are compared. The objective is to assess which strategy performs better, particularly when training data is few. To identify the best approach for text categorization, the effect of larger training sample sizes on model accuracy is also examined.

Methodology:

1.Gathering and Preparing Data:

- The IMDB dataset was downloaded, and unsupervised data was eliminated.
- Created a validation set by copying 20% of training files to a new val/ directory, retaining class balance (pos and neg).
- Datasets for training, validation, and testing were loaded using text_dataset_from_directory().
- For vectorization, a text-only dataset (text_only_train_ds) was produced.

2. Text Preprocessing:

- Limited vocabulary to 10,000 most common terms using Text Vectorization. Reduce the length of each review to 150 words.
- The training data's vectorization layer was modified.

3. Model 1: Bidirectional LSTM with Custom Embedding :

- Data input that was transferred through:
- Layer of embedding (trainable) ,
- LSTM Dropout Layer in both directions ,
- Sigmoid-activated dense output layer ,
- Combined with the rmsprop optimizer and binary_crossentropy loss.
- 10,000 samples were used for validation after 100 samples were used for training.

4. Model 2: Bidirectional LSTM Loaded GloVe 100D Word Vectors + Pretrained GloVe Embeddings:

- Built an embedding matrix that matched the language of the tokenizer.
- Constructed the model using:
- Untrainable GloVe weights are initialized in the embedding layer.
- Dense layers and LSTM dropout in both directions.
- Comparable to Model 1 in terms of compilation and training.

5. Comparison and Evaluation of Performance:

- Plotted and monitored accuracy/loss for training and validation throughout ten epochs.
- Both models were compared with increasing training sample sizes (100, 200, 500, and 1000).
- To display model performance at each level, I plotted accuracy against sample size and made summary tables.

Instruction and Verification Graphs of Performance:

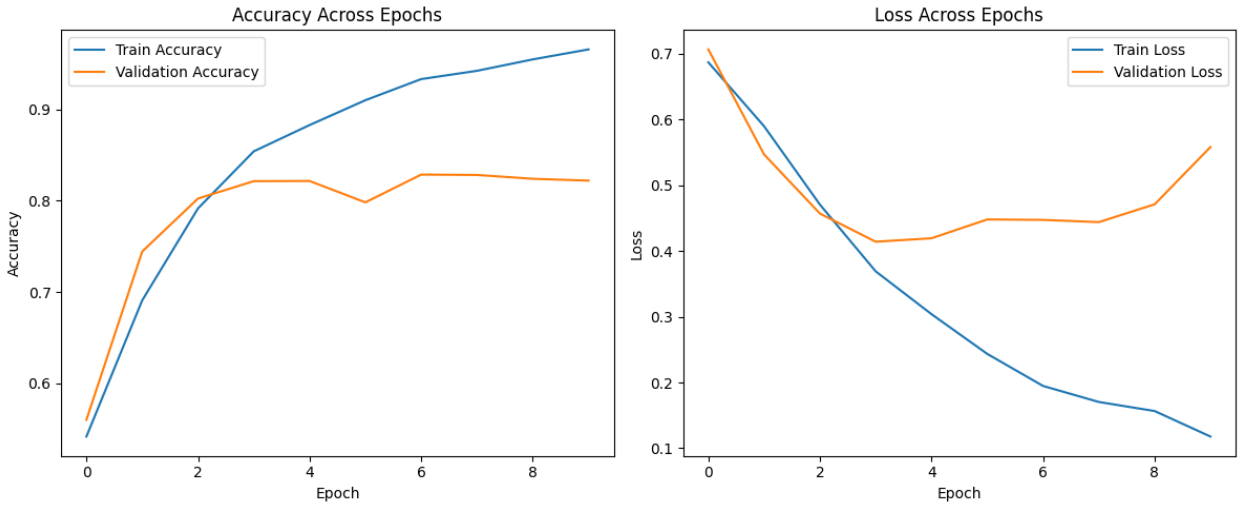
- Two models were trained:
- Model 1: LSTM + Custom Embedding
- Model 2: GloVe Embedding + LSTM Pretrained
- Training was done on 100 training samples and 10,000 validation samples across 10 epochs for both models.
- The plotted graphs were:

Validation and Training Loss and Accuracy

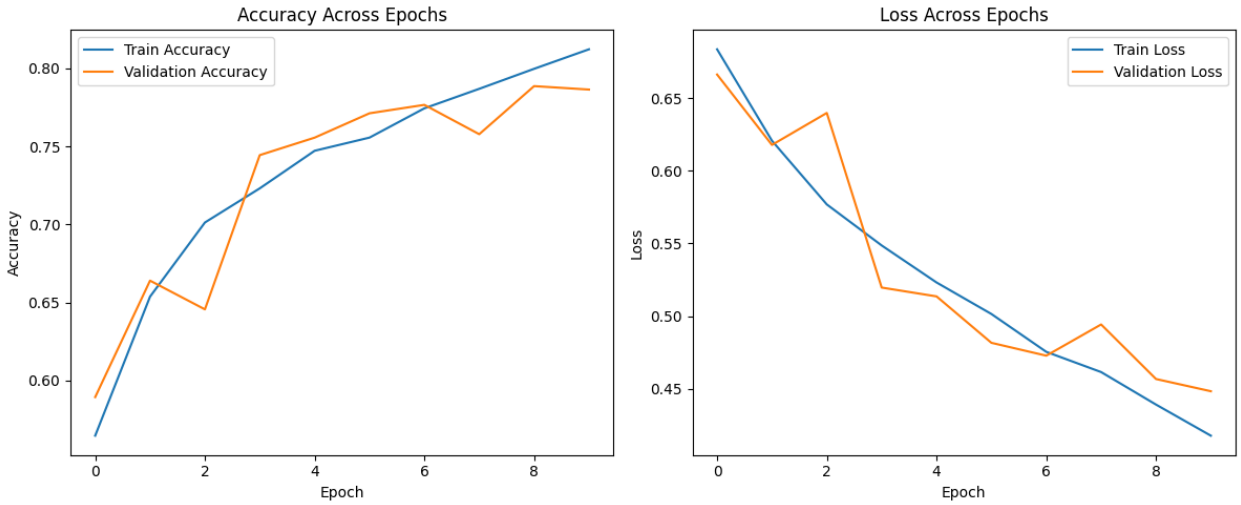
The plots for both models, which display trends in accuracy and loss throughout 10 epochs of training and validation, are shown below.

Sample Size's Impact on Model Accuracy Both models were trained on different sample sizes (100, 200, 500, and 1,000) to ascertain the effect of sample size on performance.

Graph 1: The Accuracy and Loss of Custom Embedding Models



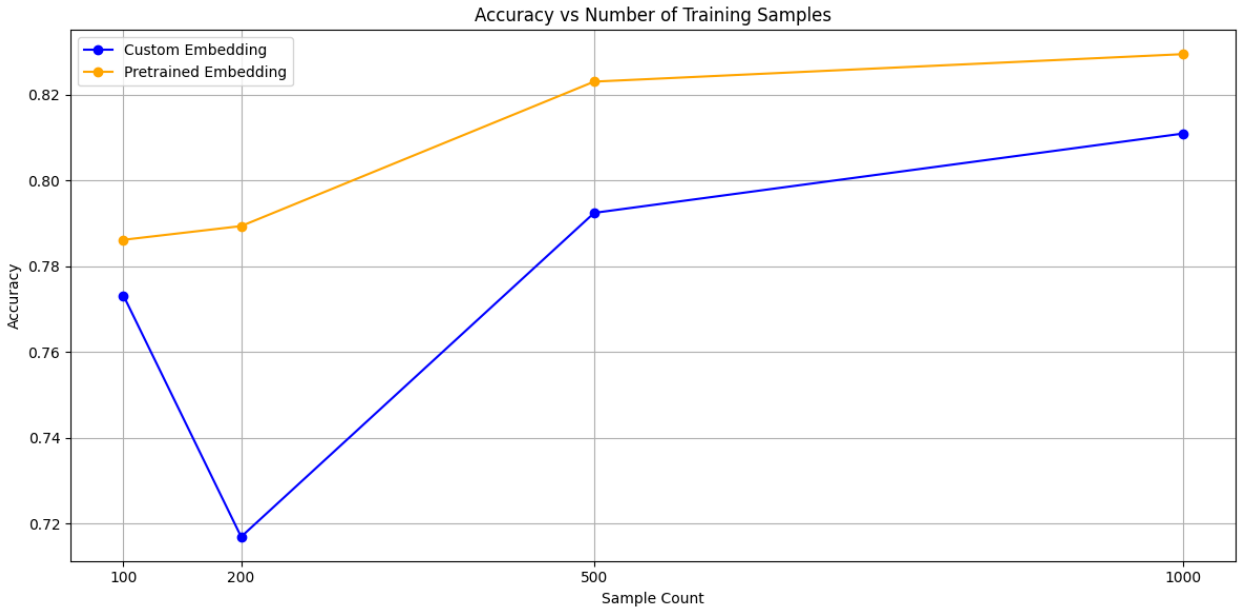
Graph 2: Accuracy and Loss of Pretrained Embedding Models



Findings:

When compared to the custom embedding model, pretrained GloVe embeddings demonstrated better validation accuracy and reduced validation loss.
Because of the reduced training set, the custom embedding model tended to overfit more quickly.

Graph 3: Plot for the Comparison: Model's Accuracy v/s Training Sample Sizes



Summary of Results:

Sample Size	Custom Embedding Accuracy	Pretrained Embedding Accuracy
100	0.77316	0.78616
200	0.71692	0.78936
500	0.79244	0.82304
1000	0.81092	0.82944

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.

Observations:

• Accuracy Difference Between Pretrained and Custom Models:
Pretrained embeddings performed marginally better than adapted embeddings with low sample sizes (100–200). At 100 samples, the displacement was +1.3%, and at 200 samples, more pronounced (+7.24%). This highlights the benefit of using pretrained embeddings when faced with limited labeled data since they bring with them preexisting semantic knowledge.
The disparity began to narrow at bigger sample sizes (500–1000), indicating that custom embeddings benefit more from task-specific training with more data.
When enough data is available, the performance difference decreased to +3.06% at 500 samples and +1.85% at 1000 samples, suggesting that bespoke embeddings can equal or even outperform pretrained ones.

• Increase in Accuracy with Sample Size:
The accuracy of both models steadily improved as the quantity of the training sample grew. From 77.31% (100 examples) to 81.09% (1000 examples), Custom Embeddings increased by about 3.78%. The pretrained embeddings were better from 78.62% (100 samples) to 82.94% (1000 samples), representing an increase of about 4.32%.
Custom models scale better with additional data and learn characteristics that are more relevant to the job, even when pretrained models have an early edge.

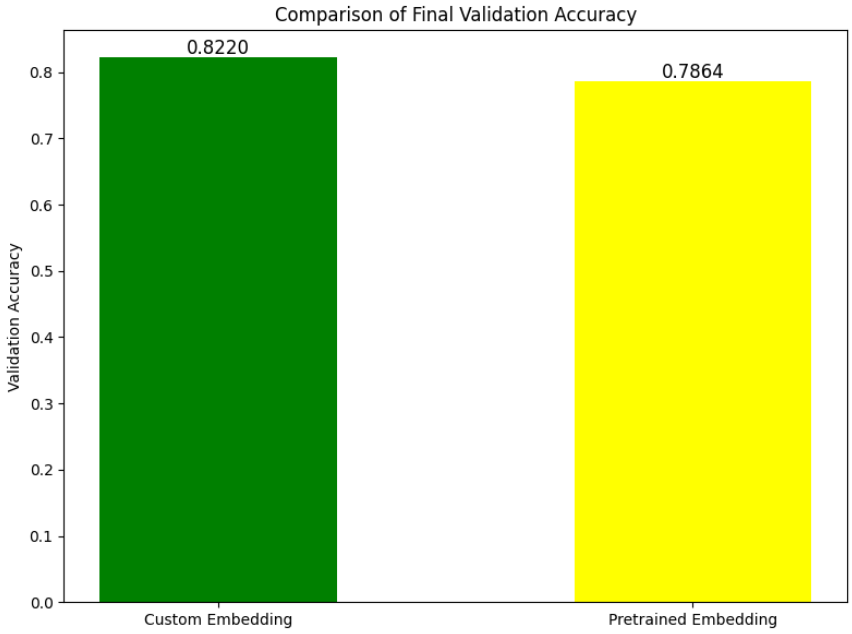
• Overall Perspective:
Low-resource environments are best suited for pretrained embeddings, but bespoke embeddings get better with more data.
The trade-off between task-specific learning (custom) and past knowledge (pretrained) is highlighted by the performance patterns.

Graph 4: Comparison of Final Validation Accuracy

After being trained on 100 samples, the accuracy of both the Custom and Pretrained models was comparable at about 78.07%.

Although the difference is small, Custom Embedding performed somewhat better than Pretrained (by 1.3%).

Conclusion: Despite having very little training data, both models function just as well.



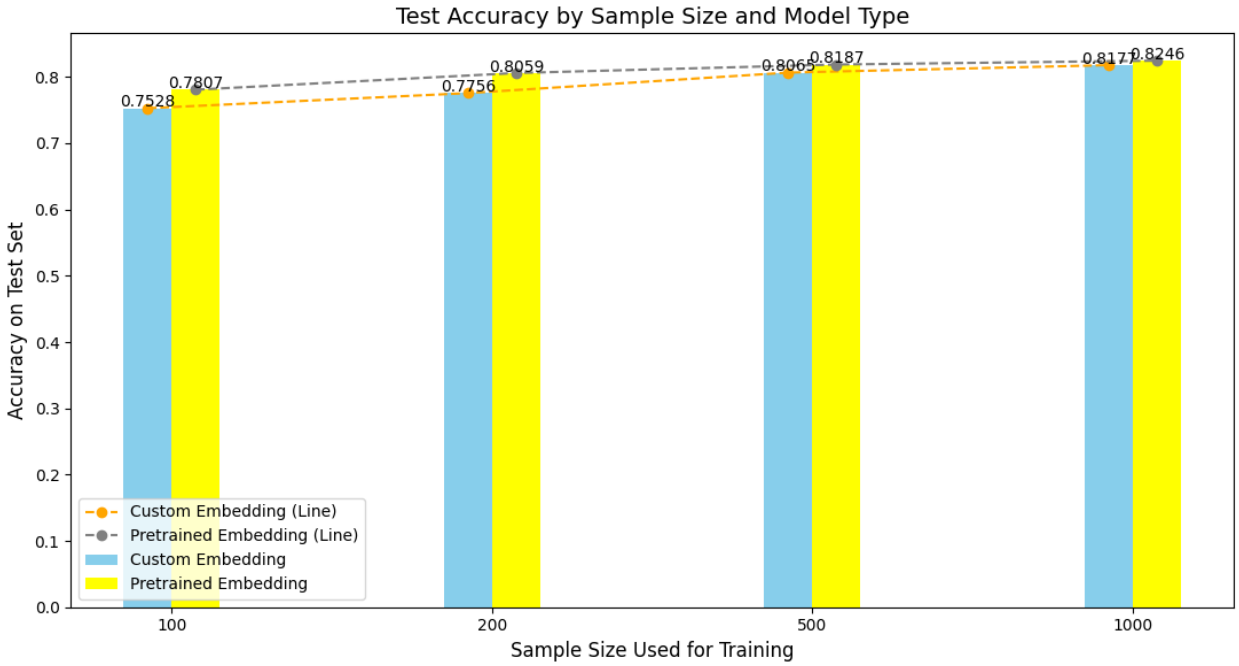
Graph 5: Accuracy of the Final Test vs Training Sample Size

Both models get more accurate as the training size grows (from 100 to 1000).

Smaller sizes (100–500 samples) yield better results from pretrained embeddings.

At 1000 samples, custom embeddings catch up with a negligible (~.85%) gap.

In conclusion, bespoke embeddings perform better with more data, whereas pretrained embeddings perform better with less data.



Key Points:

- Due to their past language expertise gained from huge corpora, pretrained embeddings perform better with smaller datasets.
- By utilizing task-specific learning, custom embeddings get much better as the training sample size grows.
- Both models performed similarly (~77.32% accuracy for Custom vs. 78.62% for Pretrained) with 100 samples, indicating that pretrained embeddings might not necessarily be the most effective in all situations.
- At 1000 samples, pretrained models had the best accuracy (82.94%), while the bespoke model came in fairly close (81.09%).
  For both models, accuracy steadily rises with sample size, demonstrating the benefit of additional training data.

Some Recommendations:

1. For small datasets, use pretrained embeddings:
When training data is scarce, use pretrained embeddings such as GloVe, which provide faster convergence and greater generalization.

2. Use Custom Embeddings with Additional Information:
To better capture task-specific subtleties, think about utilizing bespoke trainable embeddings if you have access to bigger datasets (≥500 samples).

3. To improve accuracy, increase the size of the training area:
additional data resulted in a steady improvement in accuracy; to increase model performance, try to gather or classify additional examples whenever you can.

4. Continually Check for Overfitting:
On short datasets, custom embeddings could overfit more quickly; to keep an eye on generalization, employ validation accuracy and dropout layers.

5. Try Out Different Hybrid Strategies:
For a possible performance increase on mid-sized datasets, experiment with fine-tuning pretrained embeddings rather than leaving them frozen.

6. Visualize Trends to Make Well-Informed Choices:
Keep utilizing visualizations such as accuracy vs. sample size to inform choices about dataset needs and model design.

To Wrap up:

When training data is scarce, pretrained word embeddings offer a significant benefit by assisting the model in generalizing early. Custom embeddings, on the other hand, become extremely competitive as the dataset expands and may even match or even outperform pretrained embeddings. Pretrained embeddings work well in low-resource settings for real-world applications, whereas bespoke embeddings work best in situations with enough labeled data. The best method for creating reliable NLP models is a hybrid one that varies according on task specificity and data amount.