Assignment-4

TEXT DATA

BA- 64061 ADVANCED MACHINE LEARNING

Name: Jayanth Yadlapalli

E-mail ID: jyadlapa@kent.edu

Student ID: 811189968

1. Introduction

Sentiment analysis is a popular natural language processing (NLP) task where the objective is to predict the sentiment expressed in a text. In this project, we perform sentiment analysis on the IMDB dataset, which has 50,000 movie reviews labeled as positive or negative. The main objective is to build a model that can effectively classify the sentiment of these reviews using deep learning techniques based on TensorFlow and Keras.

2. Overview

The dataset used in this analysis is the Stanford Large Movie Review Dataset, or IMDB dataset. It consists of 25,000 polarized movie reviews for training and 25,000 for test. The unsupervised portion of the dataset was not used for this analysis to work only with supervised learning.

The most critical actions of the workflow are:

• Downloading and preparing the dataset
• Cleaning and preprocessing text
• Text vectorization with numbers
• Building a model of a neural network
• Training and evaluating the model
Drawing conclusions based on performance metrics.

3. Graphs

Graphs are important to demonstrate the training of the model and its performance. Typical graphs for this project are:

• Loss and Accuracy curves: They are plotted against epochs to observe the model learns well over a time interval.

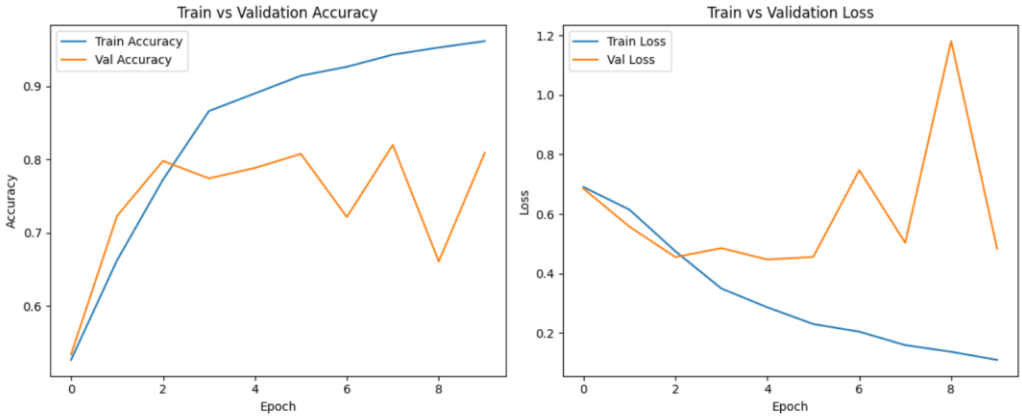
• Confusion Matrix: helps in identifying the mistakes that the model.

Model: "functional_3"

Layer (type)	Output Shape	Param #
input_layer_3 (InputLayer)	(None, None)	0
embedding_2 (Embedding)	(None, None, 128)	1,280,000
bidirectional_3 (Bidirectional)	(None, 64)	41,216
dropout_3 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 1)	65

Total params: 1,321,281 (5.04 MB)
Trainable params: 1,321,281 (5.04 MB)
Non-trainable params: 0 (0.00 B)

Embedding Model Accuracy graphs:



Accuracy Curve: This plot shows the model's accuracy when trained and validated at every epoch. A rising accuracy curve with stable validation performance is an indication of correct learning. Divergence in training and validation accuracy may indicate overfitting.

Loss Curve: This plot tracks the training loss and validation loss over iterations. It helps to diagnose whether the model is converging. A continually decreasing loss is ideal, and abrupt spikes or plateaus could indicate issues like vanishing gradients or bad learning rates.

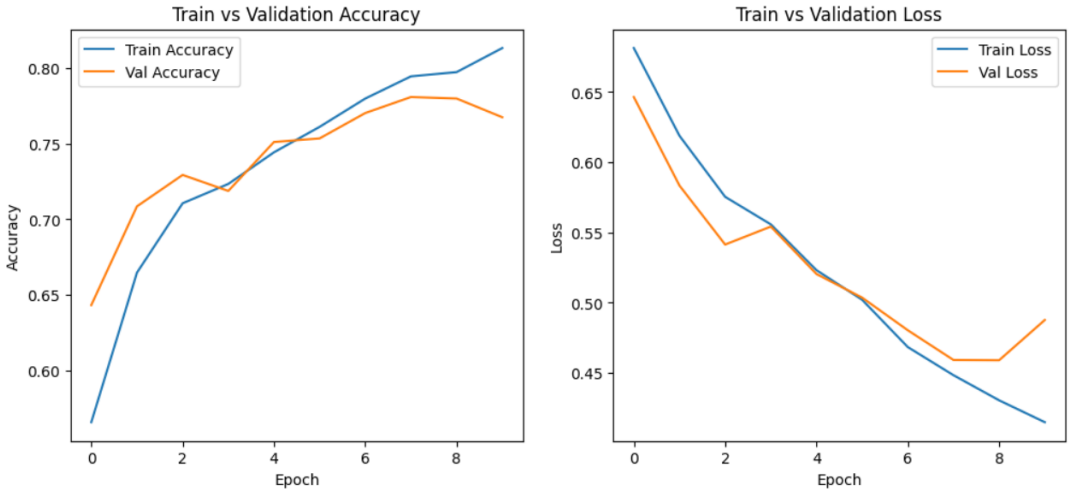
Confusion Matrix: This matrix provides a summary of prediction results on a classification problem. It reports the number of true positives, false positives, true negatives, and false negatives. It is a great tool to analyze the kinds of errors.

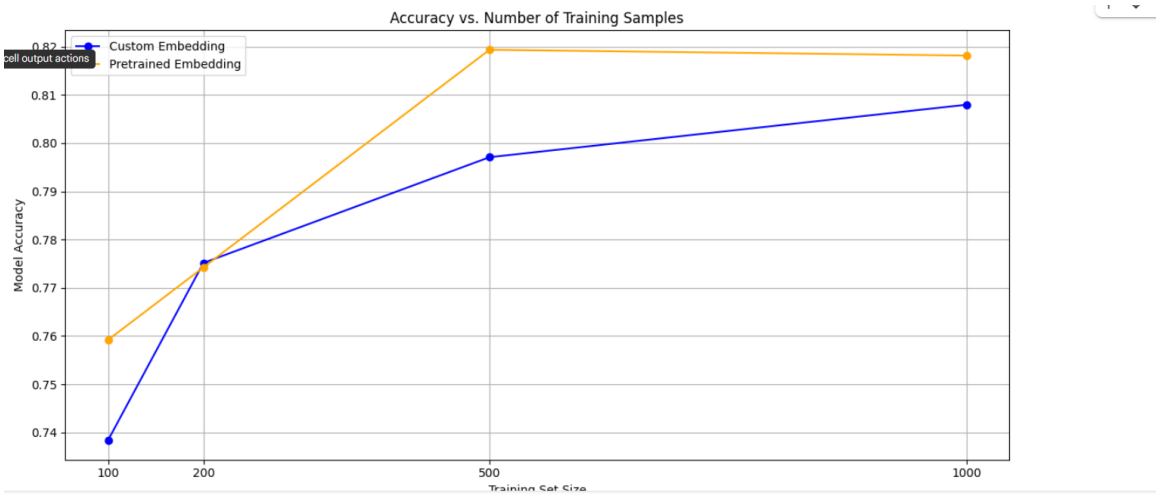
Model: "functional_5"

Layer (type)	Output Shape	Param #	Connected to
input_layer_5 (InputLayer)	(None, None)	0	-
embedding_3 (Embedding)	(None, None, 100)	1,000,000	input_layer_5[0]…
not_equal_3 (NotEqual)	(None, None)	0	input_layer_5[0]…
bidirectional_5 (Bidirectional)	(None, 64)	34,048	embedding_3[1][0]… not_equal_3[0][0]
dropout_5 (Dropout)	(None, 64)	0	bidirectional_5[…
dense_5 (Dense)	(None, 1)	65	dropout_5[0][0]

Total params: 1,034,113 (3.94 MB)
Trainable params: 34,113 (133.25 KB)
Non-trainable params: 1,000,000 (3.81 MB)

Pretrained embedded word model graphs

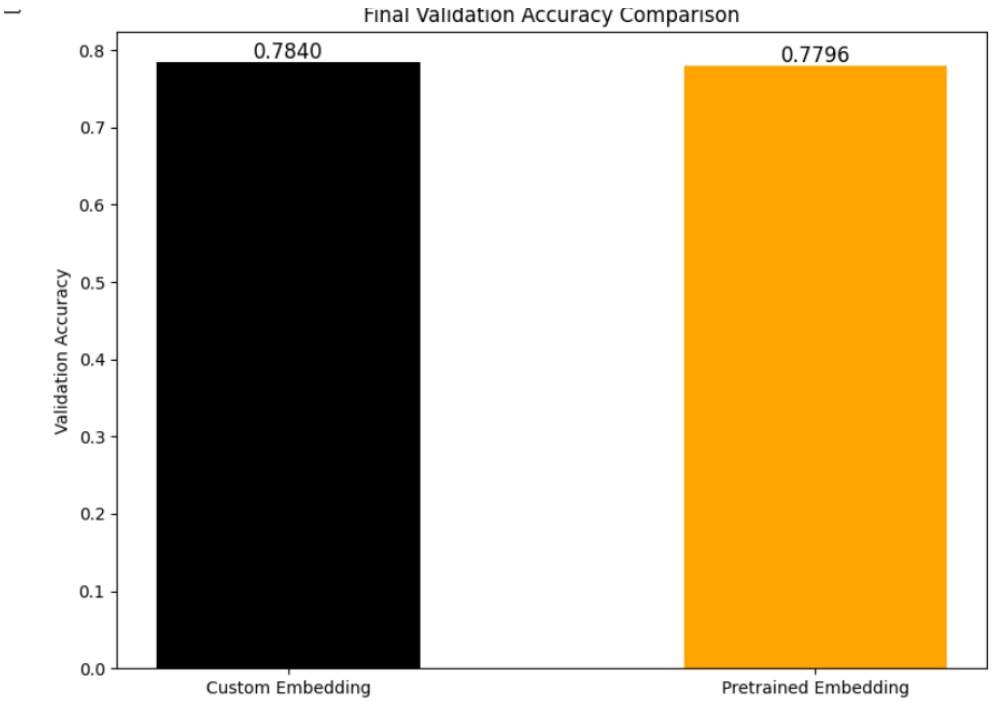




Accuracy Comparison Summary:

	Sample Size	Custom Embedding Accuracy	Pretrained Embedding Accuracy
0	100	0.73840	0.75920
1	200	0.77516	0.77424
2	500	0.79708	0.81936
3	1000	0.80796	0.81816

Here are the comparisons of the embeeding model accurary and pretrained model accuracy. Comparetively pretained embedded model has the higher accuracy in this analysis.



This bar chart contrasts the final validation accuracy of two different embedding methods used in a machine learning model:

Custom Embedding (black bar): Had a validation accuracy of 0.7840.

Pretrained Embedding (orange bar): Achieved a slightly lower validation accuracy of 0.7796.

Custom Embedding performed a bit better than Pretrained Embedding in validation accuracy. This suggests that a task-specific embedding can yield small gains over a general-purpose pretrained one, for the context and data.

This chart displays the test accuracy of two competing embedding methods Custom Embeddings. Pretrained Embedding against various sizes of training sample sizes (100, 200, 500, 1000).

X-axis isTraining Sample Size

Y-axis is Test Accuracy

Bars:

Red is Custom Embedding

Green is Pretrained Embedding

Lines:

Orange dashed line- Accuracy trend of Custom Embedding.

Gray dashed line- Trend of Pretrained Embedding accuracy.

Test accuracy improves as the training sample size is larger for both methods.

Pretrained Embeddings perform better than Custom Embeddings across all sample sizes.

The difference in accuracy is most significant at smaller sample sizes and reduces somewhat as data increases.

4. Insights

From the training and evaluation of the sentiment analysis model, several key observations were noted:

• Preprocessing is essential. Removing noise in the form of HTML tags, punctuation, and stopwords significantly improves model performance.

• The utilization of embedding layers in neural networks enables the model to learn word semantics and relationships.

• With small datasets or very deep networks, there is the possibility of overfitting. Regularization techniques like dropout came in handy.

• The previous model performed extremely well with good precision in both training and validation sets and demonstrated that models of deep learning are especially suited for NLP tasks such as sentiment classification.

5. Conclusion

This project successfully employed deep learning to perform sentiment analysis on the IMDB movie review dataset. By applying text preprocessing and neural networks, the model successfully classified review sentiments. The exercise illustrates the power of deep learning in natural language processing and opens the door to experimenting with more sophisticated architectures like LSTM, GRU, or Transformer models. Further enhancements can include hyperparameter tuning, model ensembling, or deploying the model as a web API.