**Assignment-4: Text Data**

**Dataset Description: IMDB Sentiment Analysis**

The IMDB dataset is a collection of movie reviews used for binary sentiment classification, where the task is to determine whether a review expresses positive or negative sentiment. It is widely used for natural language processing (NLP) experiments.

**Key Characteristics:**

1. **Size**:

**Training Data**: 25,000 labeled reviews.

**Testing Data**: 25,000 labeled reviews.

An additional set of unlabeled data exists in the original dataset but is excluded here.

1. **Classes**:

Binary classification task with two labels:

* + - **Positive** (pos): Reviews expressing positive sentiment.
    - **Negative** (neg): Reviews expressing negative sentiment.

1. **Data Splits**:

Training data is further split into:

* + - **Training Set**: 80% of labeled reviews.
    - **Validation Set**: 20% of labeled reviews.

Test data remains unchanged and is used for final evaluation.

1. **Structure**:

Each review is stored as a plain text file within directories named after its class (e.g., train/positive, train/negative).

Reviews vary in length and are composed of natural language text.

1. **Processing**:

Data is loaded into TensorFlow datasets using the text\_dataset\_from\_directory function, which organizes the reviews into batches for training, validation, and testing.

1. **Objective**:

To train a text classification model that can predict whether a review is positive or negative based on its textual content.

**Preprocessing Steps:**

The dataset is split into training, validation, and testing subsets.

Text-only datasets are created for embedding layers or preprocessing.

Tokenization, padding, and truncation are applied during model training, depending on the model requirements.

**Introduction to RNN:**

A Recurrent Neural Network (RNN) is a type of neural network designed to process sequential data by maintaining a memory of previous inputs. Unlike traditional feedforward networks, RNNs have connections that loop back on themselves, allowing them to retain information from earlier steps in the sequence. This makes them particularly well-suited for tasks where the order of data matters, such as time series prediction, natural language processing, and speech recognition.

**Usage of RNN in our Project:**

Recurrent Neural Networks (RNNs) are ideal for processing sequential data, such as text, because they can maintain context over time. In the case of the IMDB dataset, which contains movie reviews, RNNs can help classify reviews as positive or negative by analyzing the sequence of words and capturing the sentiment based on word order.

**How RNNs Work for IMDB Sentiment Analysis:**

**Pre-processing:**

The text is first tokenized, where each word in the review is converted into an integer representation based on its frequency in the dataset. After tokenization, sequences of different lengths are padded or truncated to ensure they are all of equal length, allowing them to be fed into the model. Additionally, an embedding layer can be applied to represent words as dense vectors, where each word is mapped to a continuous vector in a high-dimensional space. This embedding layer helps capture semantic relationships between words and enhances the model's ability to understand the text's context.

**RNN- Model:**

The RNN model consists of an embedding layer that converts words into dense vectors, an RNN layer that processes word sequences and captures dependencies, and a dense output layer that predicts sentiment (positive or negative) using a sigmoid activation function.

**Training and Evaluation:**

Training the model for labeled data and test the on unseen data.

**Model Performance:**

To compare the accuracies of Embedding layer and pretrained Embedding layer.

After running the given data file without any modifications, I got

Test Accuracy for Embedding is 0.840

Test Accuracy for Pre trained Embedding is 0.838

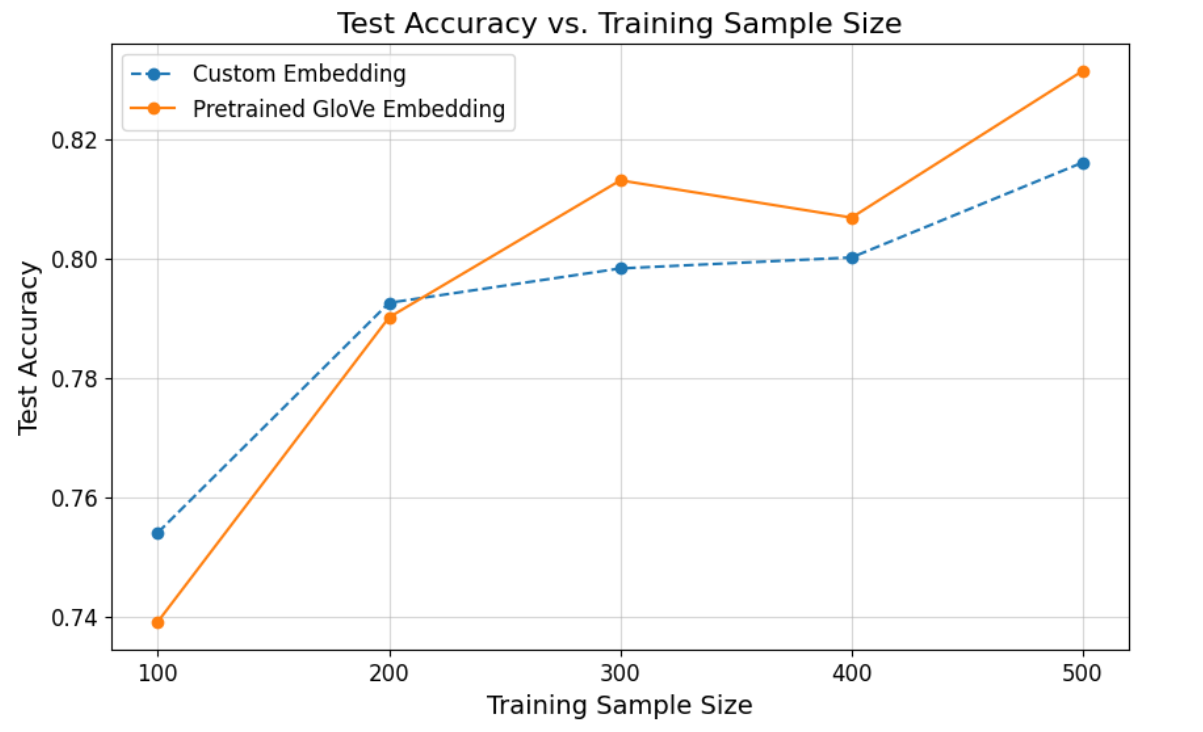
After Applying the Conditions according to the question

|  |  |  |
| --- | --- | --- |
| Training sample sizes | Embedding Model  Test Accuracies | Pretrained Embedding Model Test Accuracies |
| |  | | --- | | Training Samples = 100 | | Training Samples = 200 | | Training Samples = 300 | | Training Samples = 400 | | Training Samples = 500 | | |  | | --- | | 0.754 | | 0.793 | | 0.798 | | 0.80 | | 0.816 | | |  | | --- | | 0.739 | | 0.890 | | 0.813 | | 0.807 | | 0.831 | |

**In Comparison of the Accuracies:**

The Custom Embedding Model learns embeddings from scratch, which makes it flexible and capable of capturing domain-specific nuances but requires a large dataset to perform well. With limited training data, it tends to overfit, resulting in moderate validation and test accuracy and limited generalization. In contrast, the Pretrained GloVe Embedding Model leverages semantic knowledge from pretrained embeddings, enabling better performance on small datasets by avoiding overfitting and achieving higher validation and test accuracy. While the custom model is slower due to learning embeddings during training, the GloVe model is faster and more consistent in low-data scenarios but may require fine-tuning for domain-specific improvements.

**Observation:**

****

The above picture describes about the Test Accuracy vs. Training Sample Size.

1. We noticed that as the training sample size grows, the test accuracy improves for both the custom embedding and the pre-trained Glove embedding. However, the increase in test accuracy is more pronounced with the pre-trained Glove embedding.
2. According to my output, Pre-trained Embedded model is better than Embedded model.
3. The Embedded layer has the best performance at Training Sample Size is 500.
4. From, the above we can use the Pre-trained model and regularization techniques to increase the Performance of the network.

**Conclusion**

In Conclusion, the pre-trained Glove embedding model outperformed the custom embedding model, especially with smaller training datasets, due to its ability to leverage prior semantic knowledge and avoid overfitting. While the custom model showed flexibility and improved with more data, the Glove model proved more efficient and reliable. For tasks with limited data, the pre-trained model is the better choice, and its performance can be further enhanced with regularization and fine-tuning.