SENTIMENT ANALYSIS ON IMDB REVIEWS

AKSHAYA MAMIDIPALLI

811292765

Objective:

In the IMDB dataset, the binary classification task aims to classify movie reviews into positive and negative categories. The dataset contains 50,000 reviews; only the top 10,000 words are assessed; training samples are restricted to 100,000, 500,000, 1000, and 100,000; and 10,000 samples are used for validation. The information has been prepared. After that, the data is fed into a pretrained embedding model and the embedding layer, and a number of strategies are evaluated to gauge performance.

Data Preprocessing:

The process of preparing the dataset involves converting each review into a collection of word embeddings, where each word is represented by a fixed-size vector. A sample limit of 10,000 is applicable. Furthermore, rather than using a string of words, a unique word was represented by each of the numbers generated from the reviews. I can get the list of numbers, but the neural network's input is not appropriate for it.
It is necessary to construct tensors using the integers. It is possible to create a tensor with integer data type and form (samples, word indices) using the integer list. In order to do so, I have to use dummy words or numbers to ensure that every sample is the same length, which entails ensuring that every review is the same length.

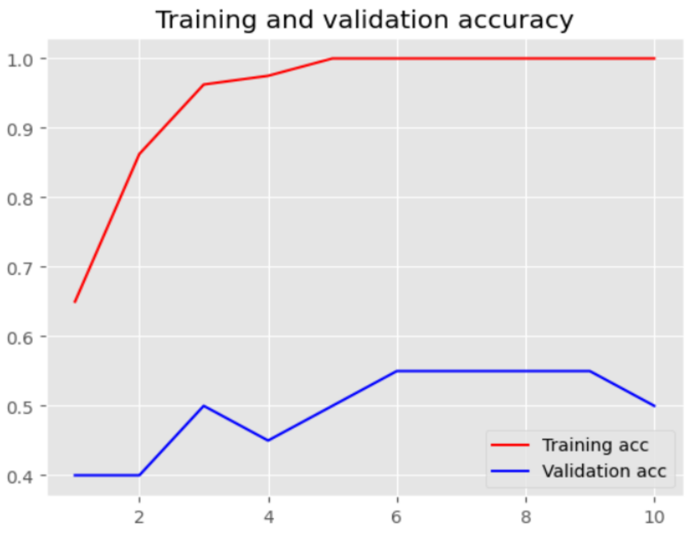
Procedure:

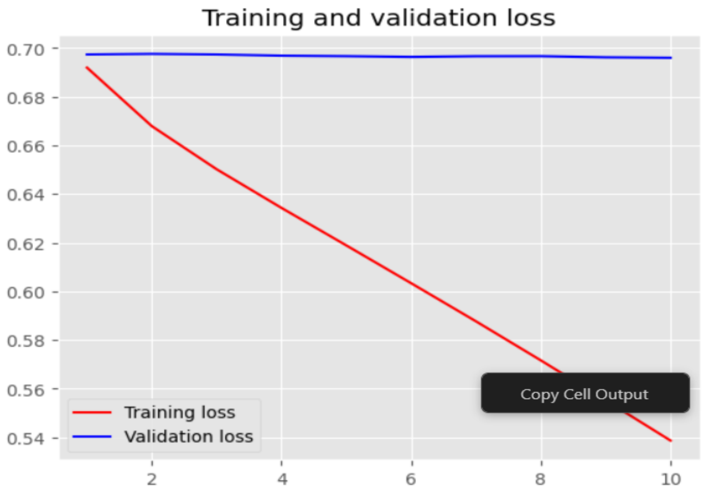
In this study, I looked into two distinct methods for creating word embeddings for this IMDB dataset:
1. An embedding layer with custom training
2. GloVe model-based pre-trained word embedding layer.
Large volumes of textual data are used to train the popular pretrained word embedding model GloVe, which we used in our work.
• To evaluate the effectiveness of various embedding techniques, I employed two distinct embedding layers—one with a pre-trained word embedding layer and the other with a custom-trained layer—using the IMDB review dataset. The accuracy of the two models was evaluated using training sample sizes of 100, 5000, 1000, and 10,000.

First, we used the IMDB review dataset to create a specially trained embedding layer. We evaluated each model's accuracy using a testing set after it had been trained on a range of dataset samples. Next, we compared this accuracy with a model that had a previously trained word embedding layer and had already been evaluated on various sample sizes.

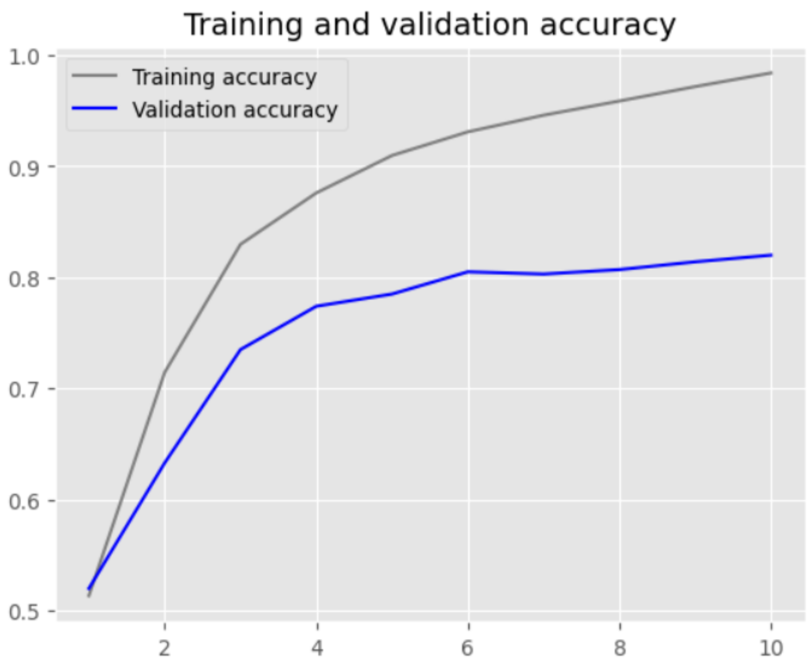
CUSTOM-TRAINED EMBEDDING LAYER

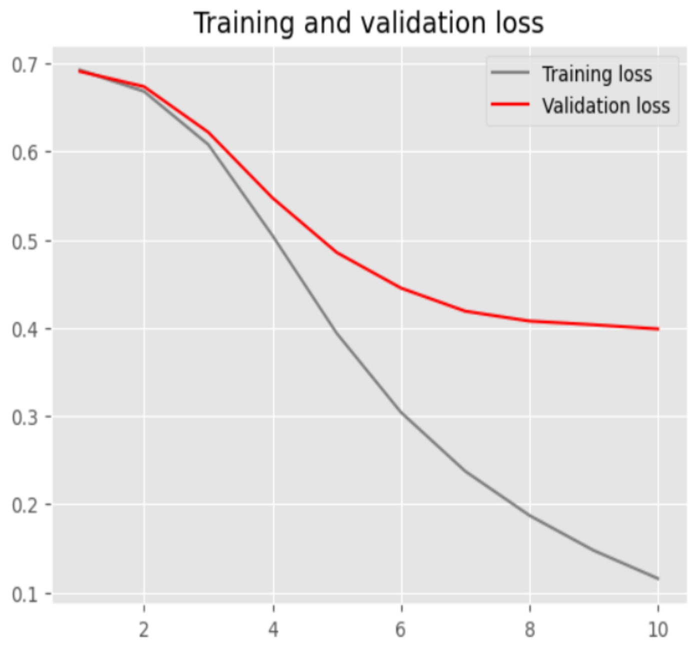
1. Custom-trained embedding layer with training sample size = 100



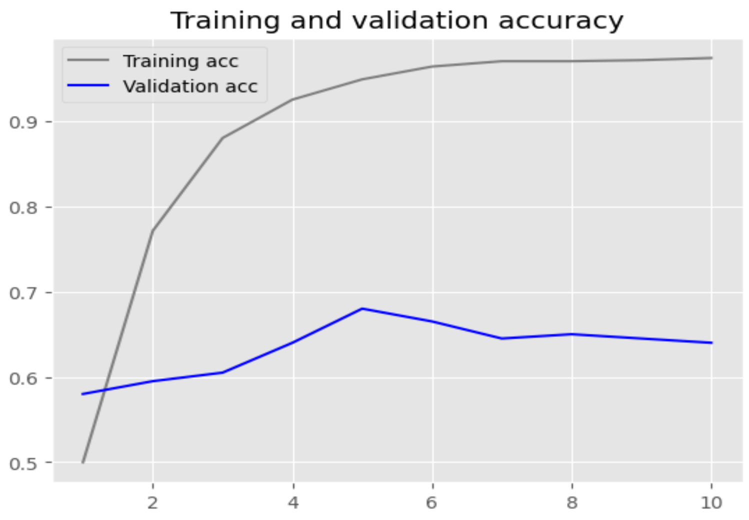


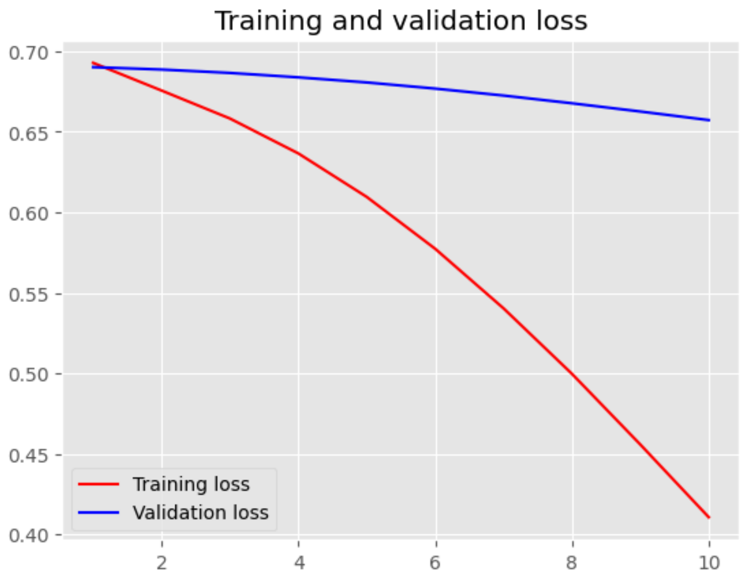
2. Custom-trained embedding layer with training sample size = 5000



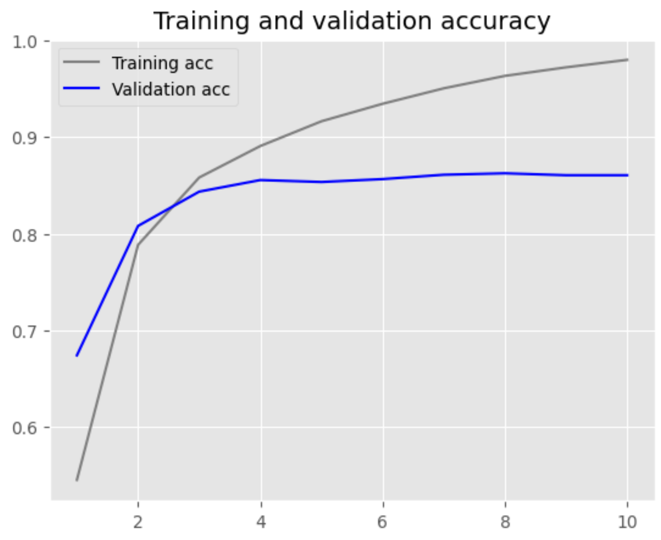


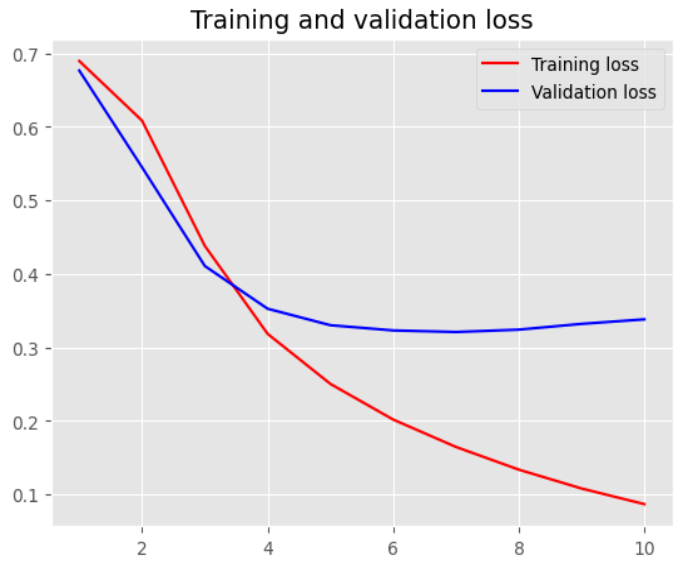
3. Custom-trained embedding layer with training sample size = 1000





4. Custom-trained embedding layer with training sample size = 10000

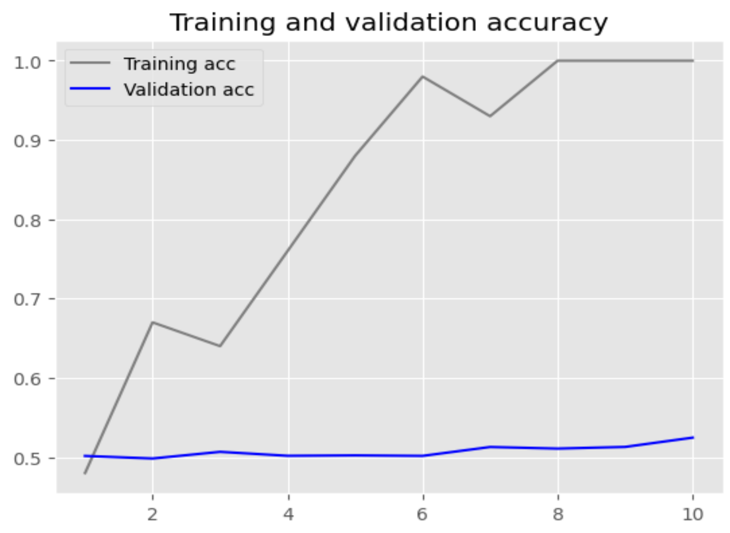


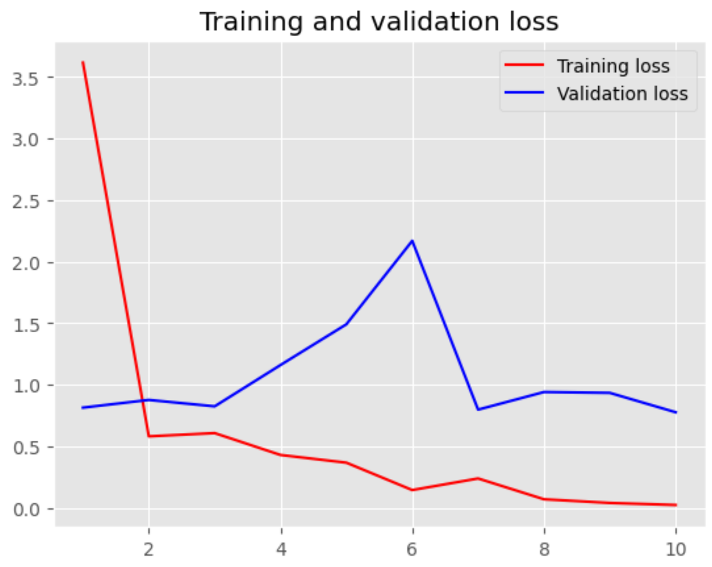


With the custom-trained embedding layer, the accuracy ranged from 97.3% to 100%, depending on the size of the training sample. The best accuracy was obtained with a training sample size of 100.

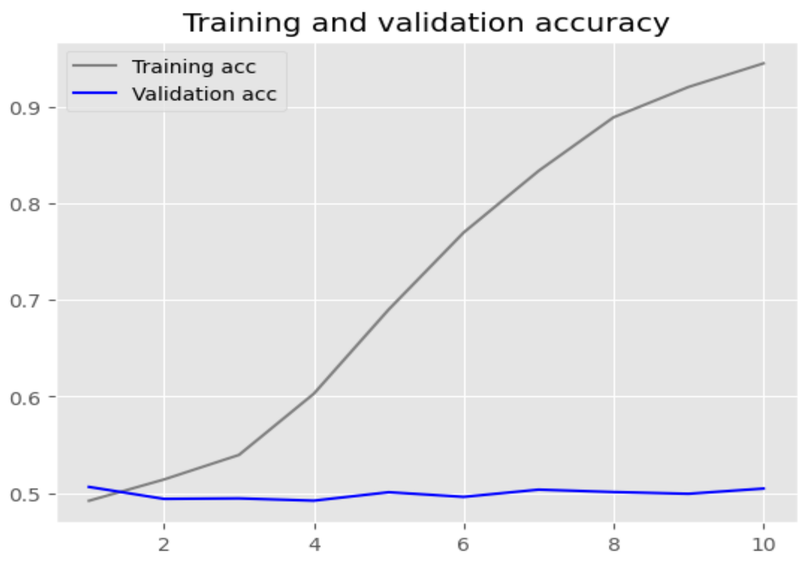
PRETRAINED WORD EMBEDDING LAYER

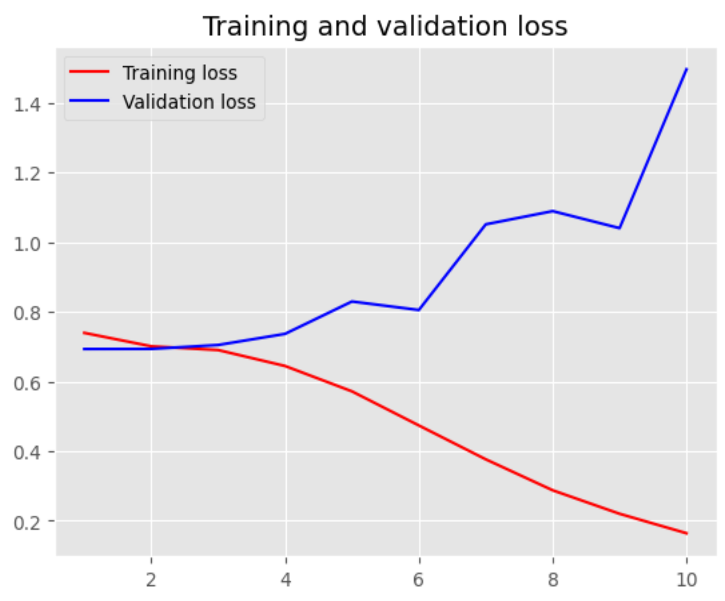
1. pretrained word embedding layer with training sample size = 100



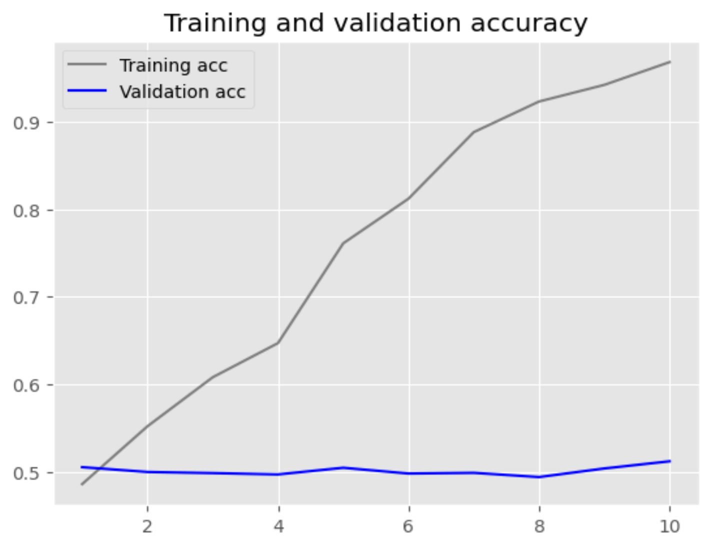


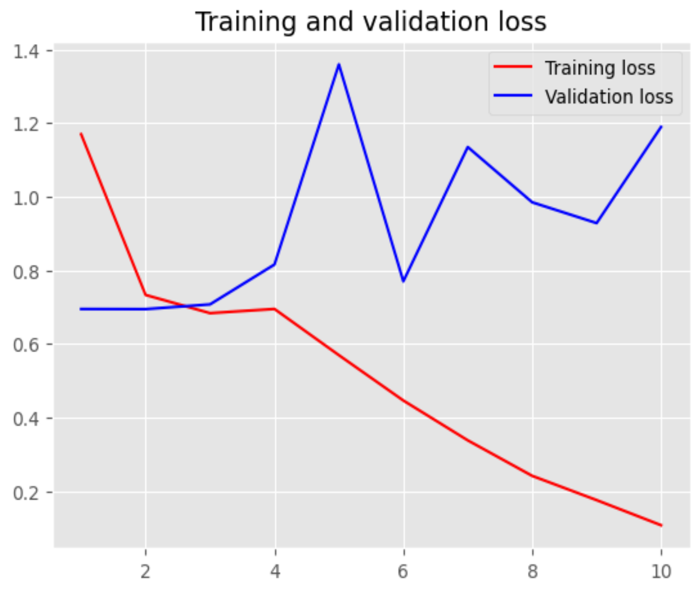
2. pretrained word embedding layer with training sample size = 500



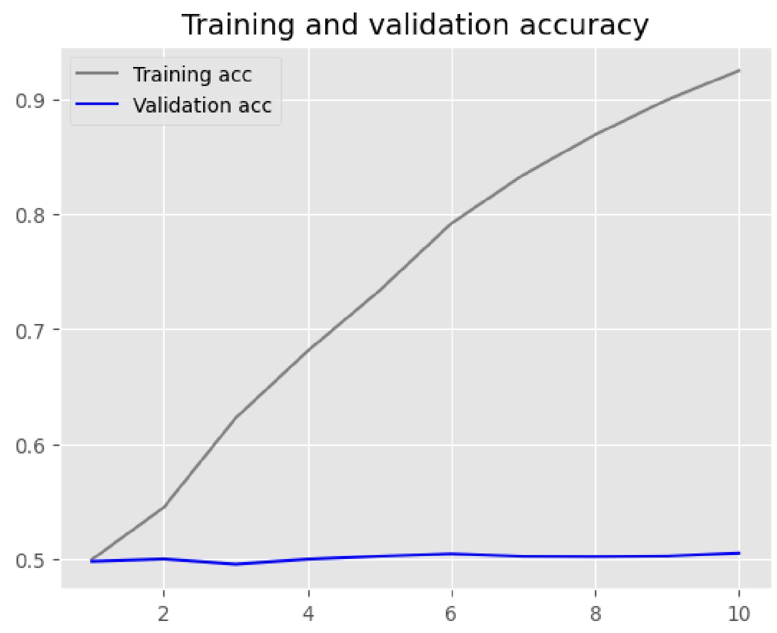


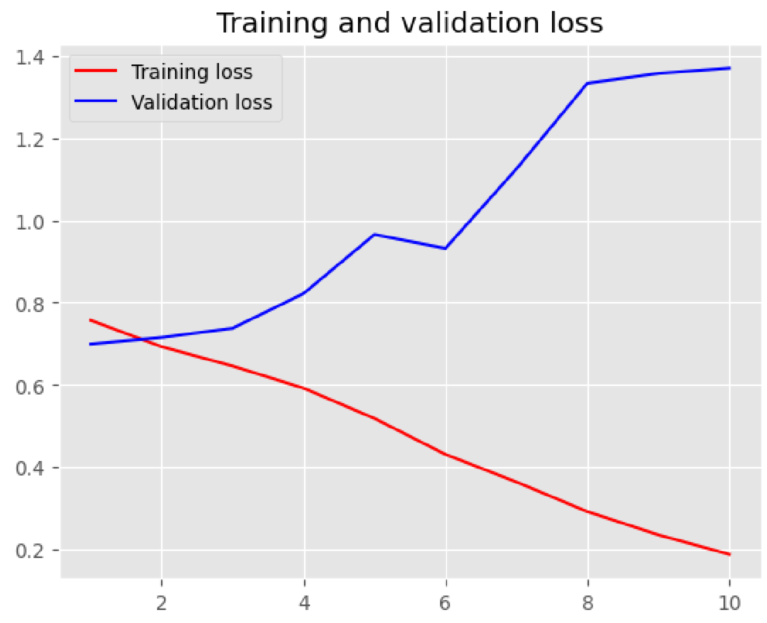
3.pretrained word embedding layer with training sample size = 1000





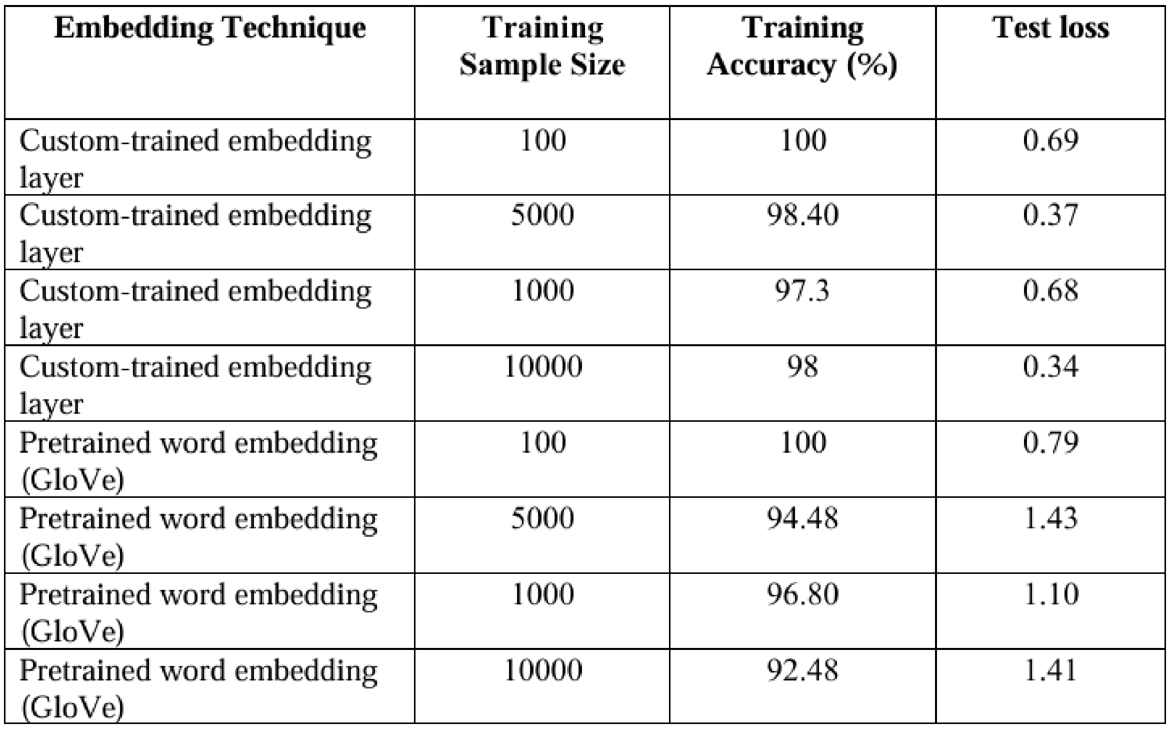
4.pretrained word embedding layer with training sample size = 10000





The pre-trained word embedding layer (GloVe) achieved accuracy ranging from 92% to 100% based on training sample size. The most precise outcome was achieved with 100 training samples. Using larger training sample sizes with pretrained embeddings causes the model to overfit, resulting in worse accuracy. Identifying the "best" strategy might be challenging due to task-specific constraints and needs.

RESULTS:



Embedding Technique	**Training Sample Size**	**Training Accuracy (%)**	**Test loss**
Custom-trained embedding layer	100	100	0.69
Custom-trained embedding layer	5000	98.40	0.37
Custom-trained embedding layer	1000	97.3	0.68
Custom-trained embedding layer	10000	98	0.34
Pretrained word embedding (GloVe)	100	100	0.79
Pretrained word embedding (GloVe)	5000	94.48	1.43
Pretrained word embedding (GloVe)	1000	96.80	1.10
Pretrained word embedding (GloVe)	10000	92.48	1.41

CONCLUSION:

However, in this experiment, the custom-trained embedding layer performed better than the pretrained word embedding layer, especially when training with more training samples. If processing power is scarce and a small training sample size is required, the pretrained word embedding layer might be a "better choice" despite the risk of overfitting.