SENTIMENT ANALYSIS ON IMDB REVIEWS

Geethika Vulli
811290653

Objective:

Sorting movie reviews into positive and negative categories is the aim of the IMDB dataset's binary classification problem. There are 50,000 reviews in the dataset; only the top 10,000 words are evaluated; training samples are limited to 100, 5000, 1000, and 100000; validation is done on 10,000 samples; reviews are examined beyond 150 words. Preparation of the data is done. Then, the data is put into the embedding layer and a pretrained embedding model, and several tactics are tested to assess performance.

- Finding the most effective strategy and if a movie review is favorable or negative is the primary objective of the binary classification job on the IMDB dataset.

Data Preprocessing:

IMDB's collection of movie reviews include classifications indicating positive and negative sentiment labels.

- Each review is converted into a set of word embeddings, where each word is represented by a fixed-size vector, as part of the dataset preparation procedure. There is a 10,000-sample limit that applies. Additionally, a series of numbers was created from the reviews, each representing a distinct word, rather than a string of words. The neural network's input is not suitable for the list of numbers, even if I have it.
- Using the numbers, tensors must be built. The integer list might be used to generate a tensor with integer data type and form (samples, word indices). For me to accomplish that, I must make sure that each sample is the same length, which means I must make sure that each review is the same length by using dummy words, or numbers.

Procedure: In this work, I investigated two different approaches to word embedding generation for this IMDB dataset:

1. Custom-trained embedding layer
2. pre-trained word embedding layer using the GloVe model.

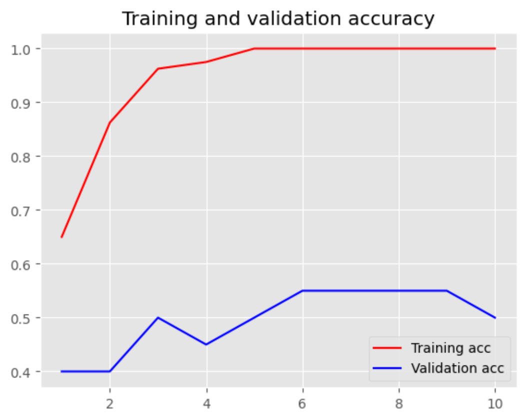
The widely used pretrained word embedding model GloVe, which we utilized in our work, is trained on large amounts of textual data.

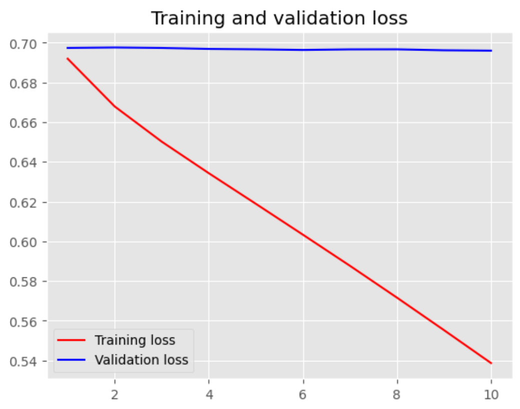
- Using the IMDB review dataset, I used two different embedding layers, one with a custom-trained layer and the other with a pre-trained word embedding layer, to assess the efficacy of different embedding strategies. I compared the accuracy of the two models with training sample sizes ranging from 100,5000,1000 &10,000.

- We began by creating a specially-trained embedding layer using the IMDB review dataset. Once each model was trained on a variety of dataset samples, we measured its accuracy using a testing set. After that, we contrasted these precisions with a model that was likewise tested on different sample sizes and included a pre-trained word embedding layer.

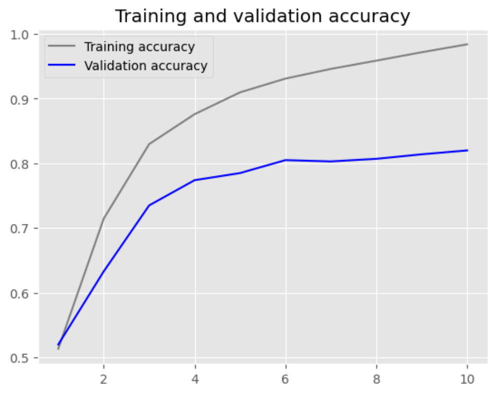
CUSTOM-TRAINED EMBEDDING LAYER

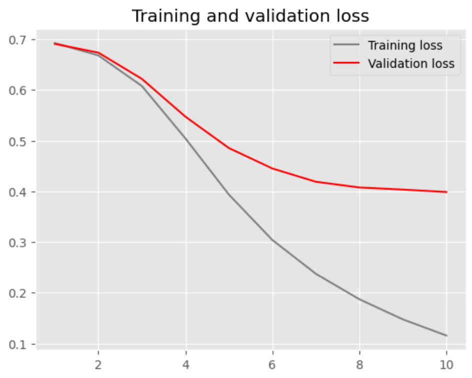
1. Custom-trained embedding layer with training sample size = 100



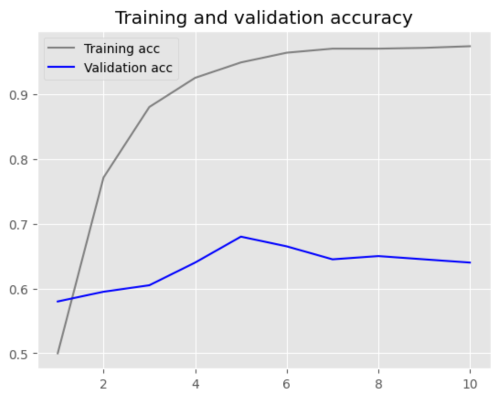


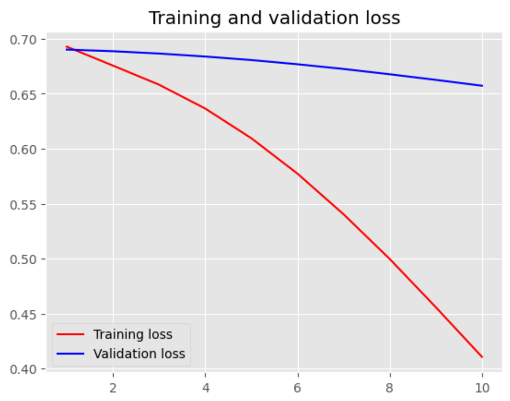
2. Custom-trained embedding layer with training sample size = 5000



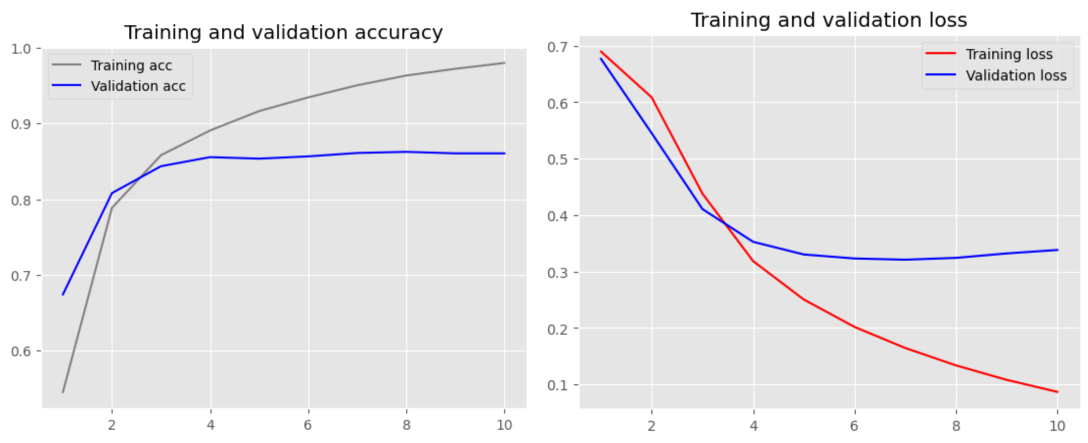


3. Custom-trained embedding layer with training sample size = 1000





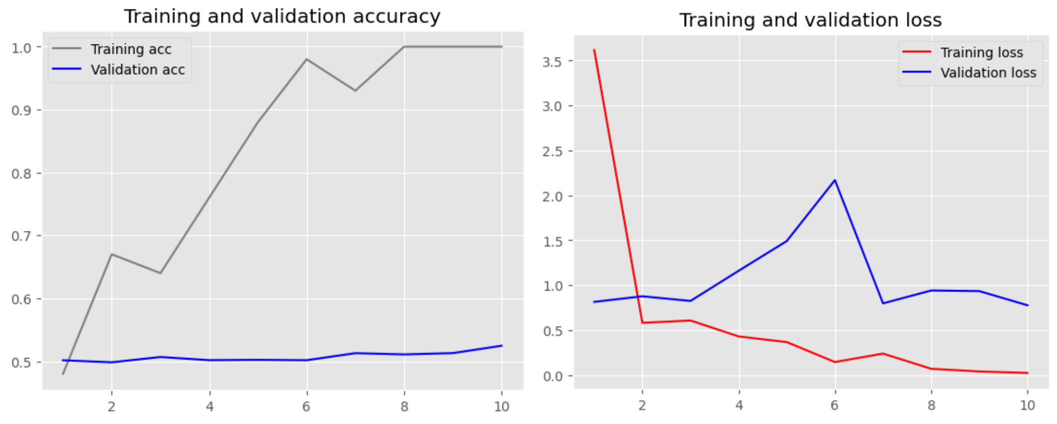
4. Custom-trained embedding layer with training sample size = 10000



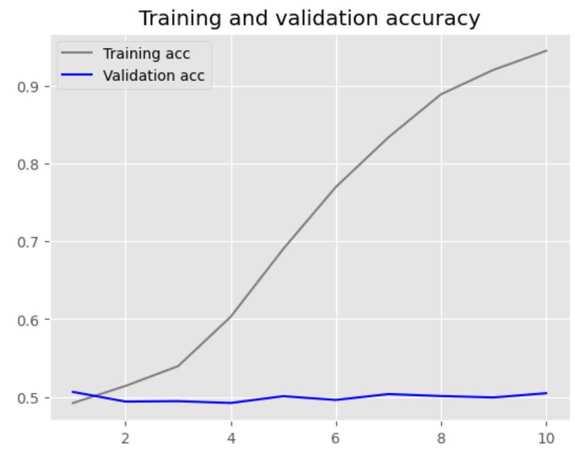
With the custom-trained embedding layer, the accuracy ranged from 97.3% to 100%, depending on the size of the training sample. The best accuracy was obtained with a training sample size of 100. It is possible that the high accuracy of this method is due to better text data representations because the embedding layer is specifically trained for the job at hand (IMDB review sentiment categorization).

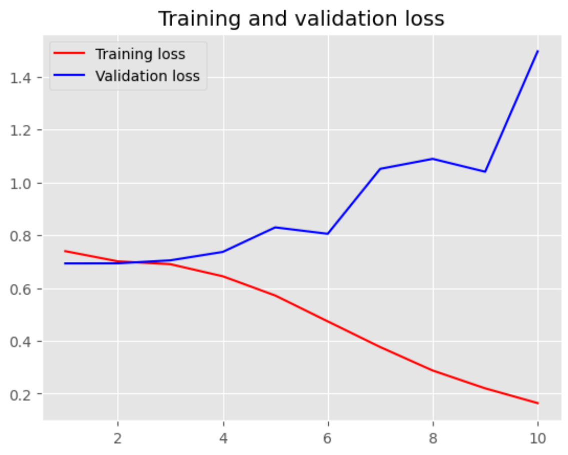
PRETRAINED WORD EMBEDDING LAYER

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

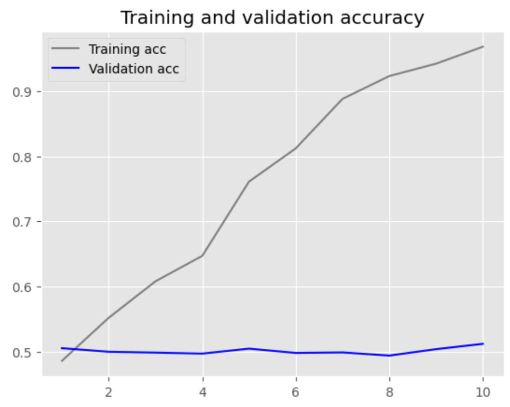


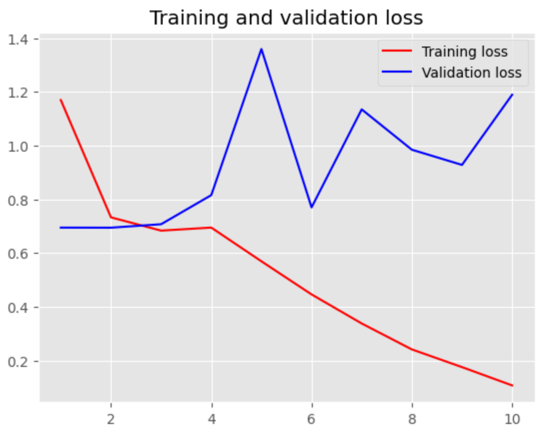
2. pretrained word embedding layer with training sample size = 5000



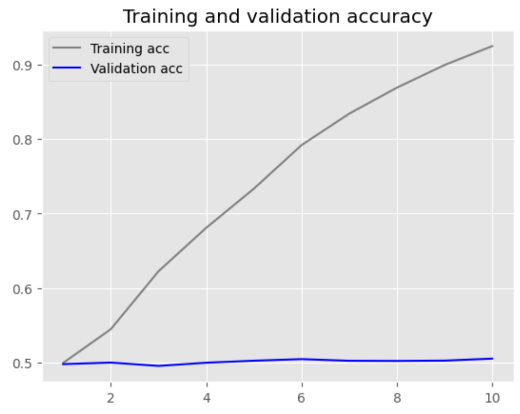


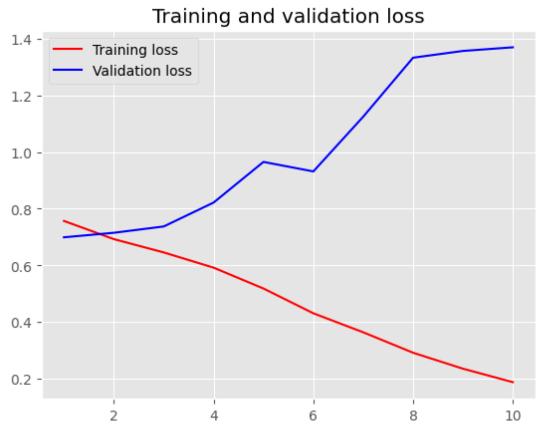
3.pretrained word embedding layer with training sample size = 1000





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





The pretrained word embedding layer (GloVe) exhibited varying degrees of accuracy, from 92% to 100%, depending on the size of the training sample. Most accurate result was obtained with 100 training samples. Because the pretrained embeddings capture a significant portion of the underlying semantic information in the text, they can be useful even with minimal training data, which may explain their excellent accuracy even with limited training samples. Lower accuracy might still arise from the pretrained embeddings' decreasing ability to capture the smallest characteristics of the job at hand as the training sample size increases. In addition, the model quickly overfits when using the pretrained embeddings with bigger training sample sizes, as the prompt mentions, which reduces accuracy. Because it depends on the needs and constraints of the task at hand, these findings make it difficult to determine which approach is the "best" to adopt with confidence.

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, the custom-trained embedding layer outperformed the pretrained word embedding layer in this experiment, particularly when training with larger training sample numbers. The pretrained word embedding layer could be a "better choice" notwithstanding the possibility of overfitting if computing resources are limited and a small training sample size is needed.