**SENTIMENT ANALYSIS ON IMDB REVIEWS**

**Pallepati Pallavi & Chaitanya Vishnu**

**811294098**

**Objective:**

The goal of the binary classification problem for the IMDB dataset is to divide movie reviews into positive and negative categories. The dataset comprises 50,000 reviews; 10,000 words out of the top 10,000 are evaluated; training samples are restricted to 100, 5000, 1000, and 100,000 samples; 10,000 samples are validated. All of the data has been prepared. After that, the data is fed into a pretrained embedding model and the embedding layer, and various strategies are tried to gauge performance.

**Data Preprocessing:**

• As part of the dataset preparation process, every review is transformed into a set of word embeddings, where each word is represented by a fixed-size vector. The 10,000 sample limit is applicable. In addition, rather than using a string of words, a set of numbers representing individual words was generated from the reviews. Even though I have the list of numbers, the input of the neural network is inappropriate for it.   
• Tensors need to be constructed using the numbers. One could create a tensor with integer data type and form (samples, word indices) using the integer list. In order for me to do that, I have to ensure that every sample is the same length, which implies that every review must be the same.

**Procedure:** For this IMDB dataset, I looked into two distinct methods for generating word embeddings:

1. A layer of custom-trained embeddings   
2. Word embedding layer that has been pre-trained using the GloVe model.   
In our work, we used the popular pretrained word embedding model GloVe, which is trained on copious amounts of textual data.   
• To evaluate the effectiveness of various embedding strategies, I used the IMDB review dataset and two distinct embedding layers, one with a custom-trained layer and the other with a pre-trained word embedding layer. I contrasted the two models' accuracy using training sample sizes of 100, 5000, 1000, and 10,000.

• Using the IMDB review dataset, we started by building a specially-trained embedding layer. After training each model on multiple dataset samples, we used a testing set to determine its accuracy. Next, we compared these precisions to a model with a pre-trained word embedding layer that had also been tested on various sample sizes.

**CUSTOM-TRAINED EMBEDDING LAYER**

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

A graph showing the performance of training and validation

Description automatically generated A graph showing a training and validation loss

Description automatically generated

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

A graph showing the performance of training

Description automatically generatedA graph showing training and validation

Description automatically generated

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

A graph showing the performance of training

Description automatically generatedA graph of training and validation loss

Description automatically generated

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

A graph showing the performance of training

Description automatically generatedA graph showing training and validation

Description automatically generated

Depending on the size of the training sample, the accuracy of the custom-trained embedding layer varied from 97.3% to 100%. The training sample size of 100 produced the best accuracy.

**PRETRAINED WORD EMBEDDING LAYER**

# pretrained word embedding layer with training sample size = 100

A graph showing the performance of training and validation

Description automatically generatedA graph showing the value of training and validation

Description automatically generated

# pretrained word embedding layer with training sample size = 5000

A graph showing the performance of training

Description automatically generatedA graph showing the loss of training and validation

Description automatically generated

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

A graph showing the growth of training

Description automatically generatedA graph showing the loss of training and validation

Description automatically generated

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

A graph showing the performance of training

Description automatically generatedA graph showing the loss of training and validation

Description automatically generated

The accuracy of the pretrained word embedding layer (GloVe) varied based on the size of the training sample, ranging from 92% to 100%. With 100 training samples, the most accurate result was achieved. Furthermore, using the pretrained embeddings with larger training sample sizes causes the model to rapidly overfit, which lowers accuracy. These results make it challenging to decide which strategy is the "best" to use with confidence because it depends on the requirements and limitations of the task at hand.

**Results:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Embedding Technique** | **Training**  **Sample Size** | **Training Accuracy (%)** | **Test loss** |
| Custom-trained embedding  layer | 100 | 100 | 0.912 |
| Custom-trained embedding  layer | 5000 | 94.32 | 1.28 |
| Custom-trained embedding  layer | 1000 | 97.7 | 1.126 |
| Custom-trained embedding  layer | 10000 | 98.12 | 0.34 |
| Pretrained word embedding (GloVe) | 100 | 100 | 0.912 |
| Pretrained word embedding (GloVe) | 5000 | 94.32 | 1.28 |
| Pretrained word embedding (GloVe) | 1000 | 97.70 | 1.12 |
| Pretrained word embedding (GloVe) | 10000 | 81.63 | 0.965 |

**Conclusion:** In this experiment, however, the custom-trained embedding layer performed better than the pretrained word embedding layer, especially when training with a higher number of training samples. If computing resources are limited and a small training sample size is required, the pretrained word embedding layer might be a "better choice" despite the risk of overfitting.