**Assignment – 4**

**Text & Sequence**

**Summary:** The task at hand is a binary classification problem using the IMDB dataset of movie reviews. The goal is to determine whether a given review is positive or negative in its sentiment. The dataset consists of 50,000 reviews, but we'll be working with a subset where only the top 10,000 most common words are considered. To analyze different model configurations, we'll train on varying sample sizes: 100, 1000, 2000, 4000 and 2,0000 reviews. We'll validate our models on a separate set of 10,000 reviews, and during evaluation, we'll truncate reviews after the first 150 words.

Before feeding the data into our models, we'll preprocess it, likely involving techniques like tokenization, padding, and encoding. Then, we'll utilize a pretrained embedding model alongside an embedding layer to convert the text into numerical representations that our models can understand. We'll experiment with different strategies and architectures, evaluating their performance on the task of classifying reviews as either positive or negative. Ultimately, our goal is to determine which approach works best for this binary sentiment classification problem on the IMDB movie review dataset.

**TECHNIQUES**

**Preprocessing of the dataset**: In the IMDB dataset, movie reviews are labeled with either a positive or negative sentiment. To preprocess this data, each review is converted into a series of word embeddings, where every word is represented by a fixed-size vector. However, we can only work with a maximum of 10,000 samples. Additionally, the reviews are transformed from a string of words into a sequence of integers, with each integer representing a unique word.

Even though we now have a list of numbers, this format is not suitable as direct input for a neural network. Instead, we need to create tensors from these integers. Specifically, we can create a tensor with an integer data type and a shape of (samples, word indices) from the list of integers. To do this, we must ensure that every sample has the same length by padding shorter reviews with dummy words (integers) until all reviews have an equal length.

So, while the reviews originally existed as text, we've preprocessed them into a numerical format that can be fed into a neural network model. By converting the words to embeddings, mapping them to integers, and creating tensors of equal length through padding, we've prepared the data for further analysis and model training.

**Approach:** For this IMDB dataset analysis, I explored two distinct approaches to generating word embeddings: using a pretrained word embedding layer with the GloVe model and training a custom embedding layer from scratch. The study utilized the GloVe model, a popular pretrained word embedding technique trained on massive text corpora. It's a go-to choose for natural language processing tasks due to its proven ability to capture syntactic and semantic relationships between words effectively. Specifically, I worked with the 6B version of the GloVe model, which was trained on a combined corpus of Wikipedia data and the Giga word 5 dataset, totaling 6 billion tokens and 400,000 unique words. Using the IMDB review dataset, I implemented two different embedding layer setups: one with a custom-trained embedding layer learned from the data, and another using the pretrained GloVe word embeddings. I then evaluated and compared the accuracy of these two models across varying training sample sizes, ranging from 100 to 10,000 reviews. First, I trained a custom embedding layer specifically on the IMDB review dataset itself, using different sample sizes. I evaluated the accuracy of models using this custom-trained embedding layer on a held-out test set. Subsequently, I assessed the accuracy of a model employing the pretrained GloVe word embedding layer across the same range of training sample sizes. So, in essence, I compared the performance of models using custom-learned embeddings versus leveraging high-quality pretrained word embeddings like GloVe, with the goal of determining the more effective approach for this particular binary sentiment classification task on movie reviews.

|  |  |  |
| --- | --- | --- |
| **Embedding**  **Technique** | **Training Sample Size** | **Accuracy** |
| Custom-trained Embedding Layer | 100 | 51% |
| Custom-trained Embedding Layer | 1000 | 75% |
| Custom-trained Embedding Layer | 2000 | 78% |
| Custom-trained Embedding Layer | 4000 | 81% |
| Custom-trained Embedding Layer | 20000 | 88% |
| Pre-trained Embedding Layer (Glove) | 100 | 69% |
| Pre-trained Embedding Layer (Glove) | 1000 | 86% |
| Pre-trained Embedding Layer (Glove) | 2000 | 88% |
| Pre-trained Embedding Layer (Glove) | 4000 | 92% |
| Pre-trained Embedding Layer (Glove) | 20000 | 99% |

Based on the analysis using RNNs on the IMDB dataset, models with embedded layers significantly outperformed other word embedding techniques in terms of both test loss and test accuracy. As we increased the sample size, the performance of the RNN models improved. When we went from training on 1,000 samples to 20,000 samples, the test accuracy increased substantially. This makes sense, as training a model on more data allows it to learn better representations and generalize more effectively.

When we compared the Custom-trained embedded layer pretrained GloVe word embeddings resulted in an even more effective model compared to training an embedded layer from scratch. The pretrained GloVe model achieved a test accuracy of 0.99% when trained on 20,000 samples, outperforming both the masked and standard embedded layer models in terms of test accuracy.

**Conclusion**

In conclusion, a model's accuracy largely depends on the amount of data it's trained on. As we increase the sample size, the model has more data to learn from, allowing it to generalize better to unseen examples. However, the optimal sample size may vary depending on the specific task and model architecture. For the given IMDB dataset, sample sizes of 5,000 and 10,000 provided good results in terms of accuracy and loss on the test set, with lower values indicating better performance. These sample sizes consistently improved performance across different embedding techniques, Custom-trained embedded layers, as well as the GloVe embeddings. The pretrained GloVe embeddings consistently showed better performance in terms of accuracy and loss across different sample sizes, outperforming the Custom-trained.

Therefore, it can be concluded that the GloVe embeddings are more efficient for sentiment analysis tasks like this IMDB dataset, compared to other embedding techniques. This is likely because they capture extensive semantic and syntactic information from large corpora, reduce the need for large training data, provide a standardized representation, and are easy to implement and use.