**REPORT ON TEXT AND SEQUENCE DATA**

**Key Insights:**

**1. Transformations or RNNs on text and sequence data:**   
Text data preparation involves the use of tokenization, padding, and embedding layers. Keras is used to generate a simple embedding-based model by converting textual data into a dense vector representation and sending it over a fully connected network.  
**2. Improving effectiveness while using less data:**   
Two common strategies for improving performance on sparse data—data augmentation and transfer learning—are not particularly covered in the previewed cells. The focus is on embedding thick layers.   
**3. Techniques to make predictions better:**   
The IMDB dataset, embedding layers, and a dense classifier are all used.   
Methods such as hyperparameter tuning, complex models (such as RNNs or Transformers), or pre-trained embeddings are not yet present in the cells being studied.

**1.** **How to apply RNNs or Transformers to text and sequence data?**

Embedding layers are used to convert text data into dense vector representations. Pre-trained embeddings (like GloVe) are used in models to set weights to enhance the processing of text data.

**2. How to improve the performance of the network, especially when dealing with limited data?**

**Pre-trained Embedding:** The model's understanding of word relationships is enhanced by using GloVe embeddings, which don't require training on large datasets. In order to preserve the previously acquired knowledge, the embedding layer is designed to be non-trainable. **Data for Validation:** A validation set is used to monitor overfitting and modify model hyperparameters.

**3. Determine which approaches are more suitable for prediction improvement?**

**RNNs:** To capture sequential patterns in text, like contextual dependencies, RNNs employ LSTMs or GRUs**.  
Switches:** Use pre-trained Transformer models (like BERT or GPT) or attention mechanisms to do text processing tasks with cutting-edge performance. **Learning via Transfer:**To get even better predictions, use fine-tuning on huge pre-trained models like BERT or RoBERTa.

**Results:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Embedding**  **Technique** | **Training**  **Sample Size** | **Training**  **Accuracy (%)** | **Test Loss** |
| Custom-trained  Embedding Layer | 100 | 100 | 0.69 |
| Custom-trained  Embedding Layer | 5000 | 97.82 | 0.39 |
| Custom-trained  Embedding Layer | 1000 | 97.16 | 0.67 |
| Custom-trained  Embedding Layer | 10000 | 97.49 | 0.34 |
| Pretrained word  Embedding (GloVe) | 100 | 100 | 1.05 |
| Pretrained word  Embedding (GloVe) | 5000 | 100 | 1.18 |
| Pretrained word  Embedding (GloVe) | 1000 | 94.42 | 1.53 |
| Pretrained word  Embedding (GloVe) | 10000 | 97.56 | 0.69 |

**Conclusion:**

In terms of training accuracy and test loss, the comparison demonstrates that custom-trained embedding layers perform better than pretrained GloVe embeddings, especially as the training sample size grows. With continuously lower test loss and good accuracy across all sample sizes, custom embeddings perform better in general. On the other hand, pretrained GloVe embeddings show inferior generalization with higher test loss but excellent training accuracy with fewer datasets. This implies that pretrained embeddings can be less successful because of possible inconsistencies with the task-specific data, whereas custom-trained embeddings are better suited for tasks with adequate training data.