**Professional Report: Comparative Analysis of RNN Models for Sequential Data Processing**

**Introduction**

Recurrent Neural Networks (RNNs) are a class of deep learning models designed to handle sequential data like text and time-series. Their architecture allows them to maintain an internal state or memory, enabling context retention during sequence processing. RNNs have demonstrated exceptional utility in tasks such as sentiment analysis, machine translation, and time-series forecasting.

This report evaluates the performance of RNN models using four different embedding techniques: **One-Hot Encoding**, **Embedded Layer**, **Embedded Masked**, and **Pre-Trained Embeddings**. The analysis is based on results from models trained on datasets of varying sizes: **Assignment\_4\_Aml\_2000**, **Assignment\_4\_Aml\_3000**, **Assignment\_4\_Aml\_5000**, and **chapter11\_part02\_sequence\_models Keerthana**. The performance is measured using test loss and test accuracy metrics, providing insights into the efficiency and reliability of each embedding technique across sample sizes ranging from 100 to 5000.

**Comparison of Accuracies and Losses**

The performance of the models is summarized in the table below. The table highlights the test loss and test accuracy for each embedding technique across sample sizes.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sample size | One hot encoded sequence | | Embedded Layer | | Embedded Masked | | Pre-Trained | |
|  | Test Loss | Test Accuracy | Test Loss | Test Accuracy | Test Loss | Test Accuracy | Test Loss | Test Accuracy |
| Initial | 0.34196 | 0.856 | 0.3570 | 0.859 | 0.3530 | 0.868 | 0.3176 | 0.867 |
| 2000 | 0.3479 | 0.848 | 0.3840 | 0.842 | 0.3683 | 0.832 | 0.3684 | 0.840 |
| 3000 | 0.3776 | 0.840 | 0.3783 | 0.843 | 0.3544 | 0.843 | 0.3696 | 0.835 |
| 5000 | 0.3588 | 0.844 | 0.3846 | 0.839 | 0.3650 | 0.835 | 0.3750 | 0.831 |

**Detailed Insights from the Results**

1. **Initial Dataset (Small Sample Size):**
   * **One-Hot Encoding** achieved moderate performance on the initial dataset, with a test loss of 0.34196 and an accuracy of 0.856.
   * The **Embedded Layer** performed slightly better, achieving an accuracy of 0.859 with a test loss of 0.3570, showing consistency across small datasets.
   * **Embedded Masked** demonstrated strong performance, achieving a test accuracy of 0.868 and a test loss of 0.3530, making it the best performer for small sample sizes.
   * **Pre-Trained Embeddings** achieved a similar level of performance to Embedded Masked, with a loss of 0.3176 and an accuracy of 0.867.
2. **Medium Dataset (2000–3000 Samples):**
   * At **2000 samples**, **One-Hot Encoding** outperformed the other techniques with a test loss of 0.3479 and an accuracy of 0.848, showing its scalability.
   * **Embedded Layer** achieved a test accuracy of 0.842 but with a slightly higher test loss of 0.3840.
   * **Embedded Masked** provided moderate performance, with a loss of 0.3683 and accuracy of 0.832.
   * **Pre-Trained Embeddings** remained competitive, showing a balanced performance with a test loss of 0.3684 and accuracy of 0.840.
   * At **3000 samples**, **Embedded Masked** and **Embedded Layer** showed comparable results, both achieving an accuracy of 0.843, with test losses of 0.3544 and 0.3783, respectively.
3. **Large Dataset (5000 Samples):**
   * For the largest dataset size, **One-Hot Encoding** continued to deliver a competitive balance, achieving a test loss of 0.3588 and accuracy of 0.844.
   * The **Embedded Layer** demonstrated consistency with a test loss of 0.3846 and accuracy of 0.839.
   * **Embedded Masked** and **Pre-Trained Embeddings** exhibited similar performance, with test losses of 0.3650 and 0.3750, respectively, and accuracies of 0.835 and 0.831.

**Key Observations**

1. **Performance by Dataset Size:**
   * **Small Datasets (<1000 samples):** **Embedded Masked** emerged as the best performer with the lowest test loss and highest accuracy.
   * **Medium Datasets (2000–3000 samples):** **One-Hot Encoding** demonstrated the best overall balance of accuracy and test loss.
   * **Large Datasets (5000 samples):** Performance differences across embedding techniques diminished, with all methods delivering comparable results.
2. **Scalability of Embedding Techniques:**
   * **One-Hot Encoding** scaled well across dataset sizes, maintaining competitive performance and low test loss.
   * **Pre-Trained Embeddings** provided consistent results across all dataset sizes, making them a reliable option for larger datasets.
3. **Limitations of Embedded Layer and Masked Embeddings:**
   * The **Embedded Layer** showed strong performance for smaller and medium datasets but struggled to maintain efficiency with larger datasets.
   * **Embedded Masked** excelled in small datasets but delivered diminishing returns as the dataset size increased.

**Conclusion**

The comparative analysis indicates that embedding techniques perform differently depending on dataset size:

* **Best Overall Performer:** **One-Hot Encoding** demonstrated consistent performance and scalability across dataset sizes, particularly excelling in medium and large datasets.
* **Small Dataset Leader:** The **Embedded Masked** method emerged as the best performer for small datasets, achieving the highest accuracy and lowest test loss.
* **Pre-Trained Embeddings:** Proved to be a strong contender for large datasets, offering a balance of computational efficiency and competitive accuracy.

**Key Takeaways**

1. **Selecting the Optimal Embedding:**
   * For **small datasets (<1000 samples):** **Embedded Masked** is the recommended choice for its robust performance.
   * For **medium datasets (2000–3000 samples):** **One-Hot Encoding** offers the best balance of accuracy and test loss.
   * For **large datasets (5000 samples):** **Pre-Trained Embeddings** are the most efficient and scalable option.
2. **Improving Model Performance:**
   * Utilize hyperparameter tuning to optimize embedding configurations for specific tasks.
   * Employ transfer learning or data augmentation techniques to boost performance on small datasets.
3. **Scalability Considerations:**
   * Performance differences among embedding techniques diminish as dataset sizes increase, making computational efficiency a key factor for large datasets.
   * Small datasets reveal more variability in embedding performance, requiring careful selection for optimal results.