# IMDB Dataset Analysis and Model Optimization

## Purpose

The objective of this assignment was to apply Recurrent Neural Networks (RNNs) and Transformers to text and sequence data, focusing on fine-tuning, validation, and performance comparison under limited data conditions. This aligns with key learning outcomes (MLOs) including understanding RNNs, Transformer architectures, and improving model performance with constrained datasets.

## Key Tasks and Results

### 1. Data Preparation and Preprocessing

- IMDB Dataset: The dataset was divided into training, validation, and testing sets. The training samples were restricted to 1,000 per class, with an 80/20 split for training and validation data.  
- Key Preprocessing Steps:  
 - Limited reviews to the top 10,000 frequent words.  
 - Each review was truncated to 150 words to maintain uniform input length.  
 - A sequence-based approach was adopted for processing text.

### 2. Model Architecture

Two main architectures were explored:  
1. Custom Embedding Layer:  
 - One-hot encoding was applied to input sequences.  
 - A Bidirectional LSTM layer was used to capture contextual information.  
 - Dropout was implemented to reduce overfitting.  
 - The output was a single sigmoid-activated dense layer for binary classification.  
2. Pre-trained Embedding Layer:  
 - Pre-trained embeddings replaced the one-hot encoding.  
 - A similar network structure with Bidirectional LSTM and dropout was used.

### 3. Fine-Tuning and Results

Custom Embedding Layer (80/20 Training Split):  
- Validation Accuracy: 95%  
- Test Accuracy: 87%  
- Fine-tuning adjustments focused on:  
 - LSTM units (32 → 64).  
 - Dropout rate adjustments.  
  
Pre-trained Embedding Layer (80/20 Training Split):  
- Validation Accuracy: 96%  
- Test Accuracy: 87%  
- Pre-trained embeddings provided faster convergence and slight improvement in accuracy.

### 4. Comparison and Insights

- Custom vs. Pre-trained:  
 - Pre-trained embeddings consistently achieved higher accuracy, especially under limited data.  
 - Validation accuracy showed greater stability with pre-trained embeddings.  
  
- Training-Validation Split:  
 - Increasing training data improved performance but introduced risks of overfitting in smaller validation sets.  
 - The 80/20 split was optimal for balancing training size and validation robustness.

### 5. Best Model and Observations

Best Result:  
- Pre-trained embedding model with 80/20 split achieved the highest validation accuracy (96%) and test accuracy (87%).

Here, I have created a table that represents the performance metrics of the various models that have been trained and evaluated for my code. Specefically, it presents the test accuracy of each model type:

A table with text and numbers

Description automatically generated with medium confidence

The experimentation reveals several insights:

1. **Model Performance**: There were notable differences in the performance of different models. The efficacy of the Embedding + LSTM model in capturing intricate relationships in text data is demonstrated by its consistent outperformance over other architectures under various variations.
2. **Effect of Data Representation**: The model performances varied depending on which data representation they used, including binary n-grams, TF-IDF n-grams, and one-hot encoding. These findings imply that model performance can be strongly influenced by the selection of data representation.
3. **Importance of Embeddings**: It is important to use pre-trained word embeddings or trainable embedding layers, as demonstrated by the Embedding + LSTM model's superior performance. Compared to simpler data representations like one-hot encoding or n-grams, this method performs better because it helps the model learn meaningful representations of words.
4. **Model Complexity**: In this task, the Transformer Encoder did not perform better than the Embedding + LSTM model, despite its complex architectural design. This implies that simpler architectures can be just as successful in sentiment analysis on the IMDB dataset as more complex models, which may not always result in better performance.