**Assignment 6**

**Problem Statement:** Sentiment Analysis Using LSTM or GRU Networks

**Library:**

* Pandas: For loading and manipulating the dataset.
* NumPy: For mathematical operations.
* NLTK: For stopword removal in text preprocessing.
* Scikit-learn: For dataset splitting into training and testing sets.
* TensorFlow/Keras: For building, training, and testing LSTM/GRU models.

**Theory:**

Sentiment analysis aims to classify textual data into sentiment categories (e.g., positive or negative). Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) are types of recurrent neural networks (RNNs) designed to handle sequential data by capturing long-term dependencies, making them ideal for tasks like sentiment analysis, where understanding context is critical.

**1.LSTM (Long Short-Term Memory)**

LSTM networks are a type of RNN specifically designed to solve the vanishing gradient problem by maintaining information over longer sequences. This is achieved through special gates within the network:

* Forget Gate: Decides which information from the previous hidden state to discard.
* Input Gate: Determines which new information to add to the cell state.
* Output Gate: Controls the output based on the current cell state and the input.

These gates allow LSTM to "remember" important information and "forget" irrelevant data across time steps, making it well-suited for sentiment analysis, especially for long text sequences where the sentiment might be influenced by words spread far apart in the input.

**2.GRU (Gated Recurrent Unit)**

GRU is a simplified version of LSTM that combines the forget and input gates into a single gate, which reduces the complexity of the model while maintaining the ability to capture long-term dependencies. GRUs are computationally more efficient than LSTMs and can often achieve similar performance, especially in tasks like sentiment analysis where speed and model simplicity might be important factors.

In a GRU:

* **Reset Gate**: Determines how much of the previous information to forget.
* **Update Gate**: Decides how much of the new information to store.

GRUs, due to their simpler architecture, tend to converge faster than LSTMs and are less prone to overfitting on smaller datasets, making them a good choice for certain sentiment analysis tasks.

**Methodology:**

1. Load and Clean Dataset: The dataset (e.g., IMDB reviews) is loaded, and non-essential parts (stopwords, punctuation) are removed.
2. Encode Sentiments: Sentiments (e.g., positive/negative) are encoded as binary labels.
3. Split Dataset: The dataset is divided into training and testing sets.
4. Tokenize and Pad Sequences: Reviews are tokenized into integers and padded to a consistent length for input into the model.
5. Build Model: A sequential model is built with an embedding layer, followed by LSTM or GRU layers, and a dense output layer.
6. Train and Test: The model is trained on the training set and tested on the test set to evaluate performance.

**Advantages:**

* Captures Long-Term Dependencies: Both LSTM and GRU can remember important information over longer sequences, improving sentiment prediction.
* Handles Sequential Data: Unlike traditional neural networks, these RNNs are designed to process sequential text data.
* Good Performance on Text: LSTM and GRU perform well in NLP tasks like sentiment analysis by learning context over time.

**Disadvantages:**

* Computationally Expensive: LSTM and GRU require more computation time due to sequential processing, especially with large datasets.
* Requires Extensive Preprocessing: Text data must be tokenized and padded, requiring more effort in data preparation.
* Sensitive to Hyperparameters: Model performance can depend heavily on the choice of hyperparameters like sequence length, hidden units, and learning rate.

**Conclusion:**

LSTM and GRU models are highly effective for sentiment analysis due to their ability to handle sequential data and long-term dependencies. While they can achieve high accuracy, they are computationally intensive and require careful preprocessing and hyperparameter tuning for optimal performance. ​​