Name: Aziz Sayyad
Roll No: 381069
Batch: A3

Assignment No: - 6

Problem Statement:
Implement sentiment analysis using Recurrent Neural Networks, specifically LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit), to classify text data (e.g., movie reviews, tweets) into positive or negative sentiment.

Objective:

- To understand the use of LSTM/GRU networks for sequential text modeling.
- To preprocess textual data effectively for deep learning models.
- To implement an LSTM/GRU-based sentiment analysis model.
- To evaluate the model using classification metrics such as accuracy, precision, recall, and F1-score.
- To visualize training and validation accuracy/loss curves to monitor overfitting and underfitting.
- To explore the impact of hyperparameters (embedding size, hidden units, learning rate, batch size) on model performance.

S/W Packages and H/W Apparatus Used:

- **Operating System:** Windows/Linux/MacOS
- **Kernel:** Python 3.x
- **Tools:** Jupyter Notebook, Anaconda, Google Colab
- **Hardware:** CPU with minimum 8GB RAM; GPU recommended (NVIDIA CUDA-enabled) for faster training
- **Libraries:** TensorFlow/Keras, NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, NLTK/Spacy

Theory:

Sentiment Analysis: Sentiment analysis is the task of identifying the emotional tone behind text data, classifying it as positive, negative, or neutral. Traditional ML models such as Naive Bayes or SVM are limited in capturing sequential dependencies. RNN-based architectures like LSTM and GRU overcome these limitations by maintaining memory across time steps.

- **LSTM (Long Short-Term Memory):**
  - Uses memory cells to store long-term dependencies.
  - Gates (input, forget, output) control the flow of information.
  - Helps prevent vanishing/exploding gradient problems in long sequences.
- **GRU (Gated Recurrent Unit):**
  - Combines the forget and input gates into an update gate.
  - Simpler than LSTM, faster to train.
  - Performs comparably in many NLP tasks while using fewer parameters.

Structure of LSTM/GRU Sentiment Model:

- **Input Layer:** Encoded text sequences (tokenized and padded).
- **Embedding Layer:** Converts words into dense vector representations (pre-trained embeddings like GloVe or Word2Vec can also be used).
- **RNN Layer:** LSTM or GRU units capture sequential dependencies. Can be stacked for better performance.
- **Dense Layer:** Fully connected layer with ReLU activation for non-linear transformations.
- **Output Layer:** Sigmoid activation (binary classification) or Softmax activation (multi-class classification).
- **Loss Function:** Binary crossentropy (for positive/negative) or categorical crossentropy (for multi-class).

Methodology / Algorithm:

1. **Data Acquisition:**
    - Collect a labeled dataset (e.g., IMDB movie reviews, Twitter sentiment dataset).
    - Ensure dataset has balanced classes to prevent bias.
2. **Data Preprocessing:**
    - Remove punctuation, special characters, URLs, and stopwords.
    - Convert text to lowercase for uniformity.
    - Tokenize text into words or subwords.
    - Convert tokens to integer sequences using Tokenizer (Keras).
    - Pad sequences to a fixed length using pad_sequences (Keras).
3. **Model Architecture:**
    - **Embedding Layer:** 100–300 dimensional word vectors.
    - **LSTM/GRU Layer:** 128–256 units. Can use return_sequences=True for stacked layers.
    - **Dropout Layer:** Prevent overfitting. Typical values: 0.2–0.5.
    - **Dense Layer:** ReLU activation.
    - **Output Layer:** Sigmoid (binary) / Softmax (multi-class).
4. **Model Compilation:**
    - Optimizer: Adam
    - Loss: Binary/Categorical Crossentropy
    - Metrics: Accuracy, Precision, Recall, F1-score
5. **Model Training:**
    - Train for 5–20 epochs depending on dataset size.
    - Use validation_split=0.1 or a separate validation set.
    - Monitor for overfitting using early stopping callbacks.
6. **Evaluation:**
    - Compute accuracy, precision, recall, and F1-score on test data.
    - Confusion matrix to visualize predictions.
7. **Visualization:**
    - Plot training and validation loss/accuracy curves.
    - Optionally, visualize word embeddings using t-SNE or PCA.
8. **Prediction:**
    - Preprocess new text input and predict sentiment using the trained model.

Advantages:

- Captures long-term dependencies in text.
- Handles vanishing gradient problem better than vanilla RNNs.
- Can work with short and long text sequences.
- Captures contextual meaning of words in sentences.
- Scalable to larger datasets and multiple sentiment classes.

Limitations:

- Computationally expensive, especially on large datasets.
- Requires large labeled datasets for good accuracy.
- Long sequences may still pose challenges.
- Sensitive to preprocessing choices.
- Difficult to interpret model decisions (black-box nature).

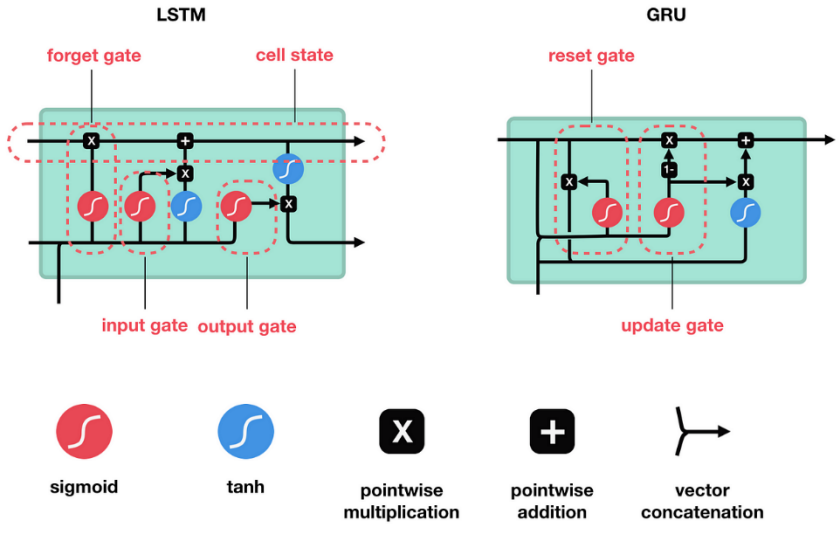
Applications:

- Customer feedback and review analysis (e.g., Amazon, TripAdvisor).
- Social media sentiment monitoring (Twitter, Facebook).
- Market research and brand perception analysis.
- Political opinion mining.
- Healthcare and patient feedback analysis.

Additional Notes / Enhancements:

- **Pre-trained Embeddings:** Using GloVe, FastText, or Word2Vec improves accuracy.
- **Bidirectional RNNs:** Capture context from both past and future words.
- **Attention Mechanism:** Enhances model by focusing on important words.
- **Hyperparameter Tuning:** Batch size, learning rate, number of units, embedding dimension can be tuned for best performance.
- **Data Augmentation:** Back-translation or synonym replacement can help with small datasets.

Diagram:



Conclusion:

Sentiment analysis using LSTM or GRU networks provides a robust solution for text classification by capturing sequential dependencies and context in sentences. Despite computational requirements, these models outperform traditional ML approaches in accuracy and semantic understanding. With proper preprocessing, hyperparameter tuning, and possibly pre-trained embeddings, LSTM/GRU models are highly effective for real-world applications in business, social media, healthcare, and political analytics.