**Name :** Chandrakant Dattatrey Thakare  
**Roll No.:**382014 **PRN:**22310303

**Class :** TY CSE AI **Batch :** B1

**Subject :** DL Lab Assignments

**Assignment 06**

**Sentiment Analysis using an LSTM or GRU Network**

**Problem Statement**

Implement a deep learning model for sentiment analysis on the IMDB movie review dataset. The model should classify a given movie review as either positive or negative. This task involves a series of steps, including data loading, text preprocessing, tokenization, sequence padding, and building a recurrent neural network (RNN) model, specifically a Long Short-Term Memory (LSTM) network.

**Objective**

* To understand the core principles of **sentiment analysis**, a key application of Natural Language Processing (NLP).
* To apply essential text preprocessing techniques, such as removing HTML tags, punctuation, and stopwords, and converting text to lowercase.
* To utilize **Keras's Tokenizer** for converting text into numerical sequences and pad\_sequences for uniform input length.
* To design, build, and train an **LSTM network** for text classification, leveraging its ability to remember long-term dependencies in sequential data.
* To evaluate the model's performance and use it to predict the sentiment of new, unseen movie reviews.

**S/W Packages and H/W Apparatus Used**

* **Operating System:** Windows/Linux/MacOS
* **Kernel:** Python 3.x
* **Tools:** Jupyter Notebook, Anaconda, or Google Colab
* **Hardware:** CPU with a minimum of 4GB RAM; GPU is optional but highly recommended for faster training.

**Libraries and Packages Used**

* **Pandas:** For data loading and manipulation.
* **NumPy:** For numerical operations.
* **NLTK:** For text preprocessing, specifically for English stopwords.
* **TensorFlow/Keras:** For building and training the deep learning model.
* **Scikit-learn:** For splitting the data into training and testing sets.
* **Re (Regular Expression):** For cleaning text data.

**Theory**

**Definition**

**Sentiment analysis**, also known as opinion mining, is a subfield of NLP that aims to determine the emotional tone or opinion expressed in a piece of text. In this assignment, it's a **text classification** task where the model learns to categorize movie reviews as either positive (1) or negative (0).

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

LSTMs are a type of **Recurrent Neural Network (RNN)** designed to handle the vanishing gradient problem, which traditional RNNs face when dealing with long sequences. They are particularly effective for tasks involving sequential data, like text, because they can retain information over long periods, understanding the context and order of words. LSTMs use a system of "gates" (input, forget, and output gates) and a "cell state" to selectively remember or forget information, which allows them to capture long-term dependencies.

**Gated Recurrent Unit (GRU)**

GRUs are a simpler, more computationally efficient variation of LSTMs. They combine the forget and input gates into a single **"update gate"** and also feature a **"reset gate"**. While they have fewer parameters and are faster to train than LSTMs, they can achieve comparable performance on many tasks. For this assignment, either architecture is suitable, but the provided code uses an LSTM.

**Methodology**

The provided Jupyter Notebook implements the following steps:

1. **Import Libraries:** All necessary libraries like Pandas, NumPy, NLTK, TensorFlow, and Scikit-learn are imported.
2. **Load and Preprocess Data:** The IMDB Dataset.csv is loaded. The load\_dataset function performs key preprocessing steps:
   * **HTML Tag Removal:** Removes <br /> and other HTML tags from the reviews.
   * **Punctuation and Character Removal:** Keeps only alphabetic characters and replaces others with spaces.
   * **Stopword Removal:** Removes common English stopwords like "the," "a," and "is" that don't contribute to sentiment.
   * **Lowercasing:** Converts all words to lowercase for consistency.
   * **Label Encoding:** Converts the sentiment labels from 'positive' and 'negative' to numerical values 1 and 0, respectively.
3. **Split Data:** The dataset is split into training (80%) and testing (20%) sets using train\_test\_split.
4. **Tokenization and Padding:**
   * **Tokenization:** The Tokenizer from Keras converts each unique word into an integer index. The training data is fit to the tokenizer, and both the training and testing sets are converted into sequences of these integers.
   * **Padding:** Since neural networks require uniform input size, the pad\_sequences function is used. It pads the shorter sequences with zeros and truncates the longer ones to a fixed maximum length, which is determined by the average length of the training reviews. This ensures all input sequences have the same dimension.
5. **Build the Model:** A **Sequential Keras model** is constructed.
   * **Embedding Layer:** This is the first layer. It takes the integer-encoded vocabulary and maps each word to a dense vector of a specified dimension (32 in the code). This layer learns a meaningful, low-dimensional representation of words.
   * **LSTM Layer:** The output from the embedding layer is fed into an LSTM layer with 64 units. This layer processes the sequences and learns the long-term dependencies crucial for sentiment analysis.
   * **Dense Output Layer:** A final dense layer with a single neuron and a **sigmoid activation function** is used. The sigmoid function outputs a value between 0 and 1, which represents the probability of the review being positive.
6. **Compile and Train the Model:**
   * The model is compiled with the **adam optimizer**, which is an efficient optimization algorithm.
   * The **binary\_crossentropy** loss function is used, which is standard for binary classification problems.
   * The model is trained for 5 epochs on the training data with a batch size of 128.
   * A **ModelCheckpoint** callback is used to save the best-performing model based on accuracy, preventing overfitting.
7. **Evaluate and Predict:**
   * The trained model's accuracy is evaluated on the unseen test data.
   * The model is then loaded and used to predict the sentiment of a new user-inputted movie review by passing the preprocessed and tokenized text through the network. A threshold of 0.7 is set to classify the review as positive.

**Advantages**

* **Improved Accuracy:** Deep learning models, especially LSTMs, excel at capturing the sequential nature of language, leading to higher accuracy compared to traditional machine learning methods.
* **Handles Context:** LSTMs can understand the context of a sentence by remembering relevant information from earlier words, which is critical for correctly interpreting complex phrases like "the movie was not bad".
* **Versatility:** The same model architecture can be applied to other sequence classification tasks, such as spam detection or topic classification.

**Limitations**

* **Computational Cost:** Training deep learning models like LSTMs can be computationally intensive and time-consuming, requiring significant resources.
* **Data Dependency:** The performance heavily relies on the quality and size of the training data. A large, well-labeled dataset is essential for a high-performing model.

**Applications**

* **Customer Feedback Analysis:** Companies can automatically analyze customer reviews and feedback to gauge satisfaction.
* **Social Media Monitoring:** Track public opinion on brands, products, and events in real-time.
* **Market Research:** Understand consumer sentiment toward new products or marketing campaigns.
* **E-commerce:** Analyze product reviews to identify areas for improvement or highlight popular features.

**Conclusion**

This assignment successfully demonstrates the implementation of a sentiment analysis system using an LSTM network. By following a structured methodology of data preprocessing, tokenization, and model training, we built a robust model capable of accurately classifying movie reviews. The project highlights the power of recurrent neural networks in handling sequential text data and their significant role in modern NLP applications.