**Assignment Report: Sentiment Analysis of IMDB Movie Reviews**

* **1. Problem Statement**
* To develop and implement a deep learning model for binary sentiment classification of movie reviews from the IMDB dataset.
* **2. Objective**
* The primary objective of this assignment is to build a sentiment analysis model using a **Long Short-Term Memory (LSTM)** neural network. The implementation involves:
* Preprocessing and cleaning a large text dataset of movie reviews.
* Converting text data into a numerical format suitable for a neural network.
* Building, training, and compiling a Sequential deep learning model with Embedding and LSTM layers.
* Evaluating the model's performance on an unseen test set.
* Creating a functional pipeline to predict the sentiment of new, user-provided reviews.
* **3. Software and Hardware Packages Used**
* **Software Packages:**
  + Python 3.8+
  + Jupyter Notebook
  + Anaconda (for environment management)
* **Hardware Packages:**
  + A standard machine with at least 8 GB RAM.
  + A GPU (e.g., NVIDIA CUDA enabled) is recommended for faster model training but not strictly required.
* **4. Libraries Used**
* **pandas:** For loading and manipulating the dataset.
* **NumPy:** For numerical operations, particularly for calculating review lengths.
* **NLTK (Natural Language Toolkit):** Used for its collection of English stopwords to filter common, non-informative words.
* **Scikit-learn:** For splitting the dataset into training and testing sets (train\_test\_split).
* **TensorFlow/Keras:** For building and training the deep learning model. Key modules include:
  + Tokenizer: To vectorize the text corpus.
  + pad\_sequences: To ensure all input sequences are of the same length.
  + Sequential, Embedding, LSTM, Dense: For constructing the neural network architecture.
  + ModelCheckpoint: To save the best performing model during training.
* **5. Theory**
* This assignment tackles a **Sentiment Analysis** task, a subfield of Natural Language Processing (NLP) that aims to determine the emotional tone or opinion expressed in a piece of text. The core technology used is a **Long Short-Term Memory (LSTM)** network, a specialized type of Recurrent Neural Network (RNN).
* **Recurrent Neural Networks (RNNs):** RNNs are designed to work with sequential data like text or time series. They maintain an internal state (or "memory") that captures information about previous elements in the sequence, allowing them to understand context. However, simple RNNs struggle with long sequences due to the **vanishing gradient problem**, where they "forget" information from earlier in the sequence.
* **Long Short-Term Memory (LSTM):** LSTMs are an advanced form of RNN that solve the vanishing gradient problem. They use a gating mechanism (input, forget, and output gates) that regulates the flow of information. This allows the network to remember relevant information over long sequences and forget irrelevant data, making them highly effective for tasks like text classification.
* **Word Embeddings:** Neural networks cannot process raw text. An **Embedding layer** is used to convert integer-encoded words into dense vectors of a fixed size (EMBED\_DIM = 32 in the code). These vectors place words with similar meanings close to each other in the vector space, allowing the model to learn and generalize semantic relationships.
* **6. Methodology**
* The implementation follows a systematic workflow from data preparation to model deployment.
* **Step 1: Data Loading and Preprocessing** The IMDB Dataset.csv is loaded into a pandas DataFrame. A comprehensive preprocessing pipeline is applied to the raw review text:
  1. **HTML Tag Removal:** Regular expressions are used to strip HTML tags (e.g., <br />).
  2. **Noise Removal:** All non-alphabetic characters are removed.
  3. **Stopword Removal:** Common English words (e.g., "a", "the", "in") are filtered out using NLTK's stopwords list, as they carry little semantic weight.
  4. **Lowercasing:** All text is converted to lowercase to ensure uniformity.
  5. **Label Encoding:** The sentiment labels are converted from strings ('positive', 'negative') to integers (1, 0).
* **Step 2: Data Splitting** The preprocessed dataset is split into a training set (80% of the data) and a test set (20%) to ensure the model can be evaluated on unseen data.
* **Step 3: Tokenization and Padding**
  1. **Tokenization:** The Keras Tokenizer is fitted on the training data to build a word index, where each unique word is assigned a unique integer. The training and test reviews are then converted into sequences of these integers.
  2. **Padding:** Since LSTMs require inputs of a uniform length, all sequences are padded or truncated to a fixed length of **130**, which was determined by the mean length of the reviews. Padding is added to the end of shorter sequences (padding='post').
* **Step 4: Model Architecture and Compilation** A Sequential model is constructed with three main layers:
  1. **Embedding Layer:** Converts the integer sequences into dense vectors of dimension 32.
  2. **LSTM Layer:** The core of the model, with 64 units to process the sequences and learn contextual patterns.
  3. **Dense Output Layer:** A single neuron with a sigmoid activation function, which outputs a probability value between 0 and 1, indicating the likelihood of the review being positive.
* The model is compiled using the adam optimizer, binary\_crossentropy as the loss function (suitable for binary classification), and accuracy as the evaluation metric.
* **Step 5: Training and Evaluation** The model is trained for **5 epochs** with a batch size of 128. A ModelCheckpoint is used to save the model with the highest training accuracy. After training, the model is evaluated on the test set, achieving an accuracy of **86.09%**.
* **Step 6: Inference** A pre-trained model is loaded to predict the sentiment of a new, user-inputted movie review. The input review undergoes the exact same preprocessing steps (cleaning, tokenizing, padding) before being fed to the model for prediction. A threshold of 0.7 is used on the model's output probability to classify the sentiment as 'positive' or 'negative'.
* **7. Advantages**
* **Context Awareness:** LSTMs can capture the order and context of words in a sentence, which is crucial for understanding sentiment.
* **Automated Feature Learning:** The Embedding layer automatically learns meaningful representations of words, eliminating the need for manual feature engineering.
* **High Accuracy:** The model achieves a solid accuracy of 86.09%, demonstrating its effectiveness for this task.
* **8. Limitations**
* **Computational Cost:** Training deep learning models like LSTMs can be time-consuming and resource-intensive.
* **Handling Nuance:** The model may struggle with complex linguistic features like sarcasm, irony, or context-dependent phrases.
* **Fixed Vocabulary:** The Tokenizer is based on the training data, so the model will not recognize words that were not present in the training set (out-of-vocabulary words).
* **9. Applications**
* Sentiment analysis has a wide range of real-world applications, including:
* **Social Media Monitoring:** Tracking public opinion on products, brands, or events.
* **Customer Feedback Analysis:** Automatically sorting customer reviews, emails, and support tickets into positive and negative categories.
* **Market Research:** Gauging audience reactions to marketing campaigns or new products.
* **Financial Markets:** Analyzing news and social media sentiment to predict stock market trends.
* **10. Output**
* The final output of the project is a trained model capable of classifying a given movie review as either **'positive'** or **'negative'**. The model's performance on the test dataset resulted in **8609 correct predictions** and **1391 incorrect predictions**, yielding a final accuracy of **86.09%**.
* **11. Conclusion**
* This assignment successfully demonstrates the entire lifecycle of building a sentiment analysis model using an LSTM network. Through effective text preprocessing and a well-structured deep learning architecture, the model proved capable of classifying movie reviews with high accuracy. This project highlights the power of recurrent neural networks for NLP tasks and provides a practical foundation for solving real-world text classification problems.