Assignment III

In this assignment, we will work with the Goodreads reviews dataset (link here) to train a model that will predict whether a review contains a spoiler or not.

Section 1: Implementing and Training an LSTM Network

Implement an LSTM network to classify text snippets as containing spoilers or not.

Tasks:

1. **Data Preprocessing:** Load the dataset, inspect the data, and preprocess it. This might involve tokenizing the text, padding sequences, and splitting the data into training and testing sets.
2. **Build the LSTM Model:** Design an LSTM model suitable for this problem.
3. **Train the Model:** Train the LSTM model on the training set, using a suitable optimizer, loss function, and performance metric. Use validation data to monitor the model's performance and adjust hyperparameters as needed.
4. **Evaluate and Interpret Results:** Report on the model's performance using metrics such as accuracy, precision, recall, and F1-score. Discuss potential areas for improvement.

Section 2: Exploring the GRU Architecture (Theory)

In this section, we will explore the theoretical foundations of the Gated Recurrent Unit (GRU) and understand how it differs from the LSTM architecture.

Tasks:

1. **Research GRU Architecture:** Explain the structure of a GRU cell, including its key components—the update gate and the reset gate.
2. **Compare GRU and LSTM:** Analyze the differences between GRU and LSTM cells. Discuss the scenarios in which GRUs might be more efficient than LSTMs and any trade-offs involved.

Note: This section is theory-based. Be thorough in your discussion and include diagrams if necessary to visualize the GRU architecture.

Section 3: Implementing and Training a GRU Network

In this section, we will implement a GRU model to classify spoilers and compare its performance with the LSTM model.

Tasks:

1. **Build the GRU Model:** Design a GRU model using the same dataset and preprocessing steps from Section 1. The model should follow a similar structure to the LSTM model, but with GRU layers replacing the LSTM layers.
2. **Train and Evaluate the GRU Model:** Train the GRU model with the same settings as the LSTM model. Evaluate its performance on the test set, using the same metrics for comparison.

3. **Comparison with LSTM:** Compare the performance of the GRU and LSTM models, and discuss which model performed better. Consider factors like accuracy, training time, and model complexity. Make sure to test your model in the same test set that you used in section 1.
4. **Discussion:** Reflect on the pros and cons of each model based on your findings.

Section 4: Exploring and Testing Optimization Algorithms for Recurrent Neural Networks
In this section, we will explore popular optimization algorithms for training neural networks, and test each algorithm to understand its impact on model performance and training dynamics. This section will combine theory and practice.

Tasks:

Part A: Theory – Understanding Optimization Algorithms

1. **Research Key Algorithms:** Review the following popular optimization algorithms that are commonly used to train RNNs:
    - **Stochastic Gradient Descent (SGD)**
    - **SGD with Momentum**
    - **RMSprop**
    - **Adam (Adaptive Moment Estimation)**
2. **Purpose and Mechanism:** For each algorithm, briefly explain its purpose in optimizing neural network models and describe how it adjusts weights to minimize the loss function
3. **Advantages and Disadvantages:** Summarize each algorithm's strengths and limitations, especially with respect to convergence, stability and computational cost

Part B: Practice – Testing Optimization Algorithms

1. **Experiment Setup:** Using the LSTM model from Section 1 or the GRU model from Section 3, experiment with each of the optimization algorithms researched in Part A. Keep other hyperparameters constant, such as batch size and learning rate, to isolate the impact of each optimizer.
2. **Train and Record Results:** Train the model using each optimization algorithm separately on the spoiler dataset. For each trial:
    - Track metrics such as accuracy, loss, training time, and number of epochs to convergence.
    - Note any differences in training stability and learning rate adjustments across optimizers.
3. **Comparative Analysis:** Summarize the observed differences in model performance and training dynamics for each algorithm. Discuss how each optimizer performed in terms of convergence speed, final accuracy, and computational efficiency. Highlight any trade-offs or notable patterns between the algorithms.