Building and Understanding a Sentiment Analysis Model with RNN

Scenario:

Imagine you are hired as a Machine Learning Engineer for a startup developing a platform to automatically analyze movie reviews from social media and determine whether the public sentiment is positive or negative. Your task is to:

- Build a text classification model.
- Justify the architectural choices you make.
- Experiment with different settings.
- Interpret the performance of your model.

Part A – Conceptual Questions

Answer the following questions in your own words:

1. Why is it necessary to use an embedding layer before feeding data into an RNN?
2. Explain how RNN processes sequences step-by-step. How is it different from a CNN?
3. Why do we use pad_sequences before passing the data into the model? What happens if we don't?
4. Discuss the role of the SimpleRNN layer in the model. How does it help in sentiment classification?
5. Why is binary crossentropy used as a loss function? Would mean squared error be appropriate? Why or why not?
6. Suppose we increase maxlen from 200 to 500. What would be the effect on:
    - Model training time?
    - Memory usage?
    - Model performance?

Part B – Practical Implementation

Perform the following in a Jupyter notebook or Colab and explain each step:

Task 1: Data Preparation

- Load the IMDB dataset using Keras.
- Explore the data: number of samples, average review length, label distribution.
- Pad sequences to a length of 200 words.
- Visualize a few original and padded sequences.

Task 2: Model Building

- Build a model using the following layers:
    - Embedding(input_dim=10000, output_dim=32, input_length=200)
    - SimpleRNN(32)
    - Dense(1, activation='sigmoid')
- Compile the model using rmsprop, binary_crossentropy, and accuracy.

Task 3: Model Training and Evaluation

- Train the model with validation_split=0.2, epochs=5, batch_size=128.
- Plot training and validation accuracy/loss curves.
- Evaluate the model on the test set and report the accuracy.

Part C – Experimental Tasks

Perform the following experiments and explain your observations:

1. Change the SimpleRNN units from 32 to 64. What changes do you notice in accuracy and training time?
2. Replace SimpleRNN with LSTM and compare the performance.
3. Try reducing the vocabulary size to 5000 and 2000. How does this affect model performance and training time?
4. Use dropout=0.2 in the RNN layer. Did it help reduce overfitting?

Part D – Reflection (Short Essay, 200–300 words)

Reflect on the entire process. What challenges did you face? What did you learn about processing sequential text data with neural networks? If you had to improve this model for a real-world production system, what steps would you take?