EXPERIMENT 4

EXPERIMENT OBJECTIVE

To explore Text Generation using Recurrent Neural Networks (RNNs) and understand the impact of different word representations:

1. One-Hot Encoding
2. Trainable Word Embeddings

Additionally, to train both the RNN and LSTM models on a Dataset of 100 poems and compare their performances.

DATA PREPROCESSING

Loading the Dataset

- The dataset is loaded from a CSV file and merged into a single text corpus.
- All lines are joined together to create a continuous stream of words..

Tokenization and Vocabulary Creation

- The text is split into individual words (tokens).
- A vocabulary is built, mapping each unique word to a specific index.

Generating Training Sequences

- One-hot encoding is applied to tokens for the **One-Hot Encoding model**.
- Indexed sequences are created for the **Embedding model**.
- Input sequences of 10 words are formed, with the next word as the target output.

NEURAL NETWORK IMPLEMENTATION

One-Hot Encoding RNN Model

Architecture

- **Input Layer:** One-hot encoded word vectors.
- **Recurrent Layer:** An RNN layer processes the input sequences.
- **Fully Connected Layer:** Maps the RNN output to the vocabulary size for prediction.

Weight Initialization

- Weights are initialized randomly.

Activation Functions

- **Softmax:** Applied to the output layer for probability distribution.

Regularization

- **None**: No regularization techniques were applied.

LSTM Model with Trainable Embeddings

Architecture

- **Embedding Layer:** Maps word indices to dense vectors.
- **LSTM Layer:** Captures long-term dependencies in the text.
- **Fully Connected Layer:** Maps the LSTM output to the vocabulary size for prediction.

Weight Initialization

- Weights are initialized randomly.

Activation Functions

- **Softmax:** Applied to the output layer for probability distribution.

Regularization

- **None**: No regularization techniques were applied.

TRAINING CONFIGURATION

Training the Model

- **Loss Function:** Cross-entropy loss.
- **Optimizer:** Adam optimizer with a Learning Rate of 0.001.
- **Epochs:** 100.
- **Batch Size:** 32.
- **Training Process:**
    - Batches of sequences are processed.
    - For the **One-Hot Encoding RNN**, sequences are one-hot encoded during training.

- For the **LSTM with Embeddings**, sequences are passed through an embedding layer.
- Forward and backward propagation is performed.
- Weights are updated using the optimizer.

TRAINING AND RESULTS

Key Performance Metrics

- The **One-Hot RNN Model** started with a higher loss and required longer training to improve.
- The **LSTM Model with Embeddings** trained faster and achieved a lower final loss.

Evaluation Results

- The **One-Hot RNN** model reached a final loss of **0.0108** after **100 epochs** but required **~1674.02 seconds** to train.
- **The Embedding LSTM** model achieved a final loss of **0.0003** after **100 epochs** and required only **~877.62 seconds** to train.

POEM GENERATION

Process

- A function is implemented to generate poetry from a given seed text.
- The model predicts the next word based on previously generated words.
- The generated poem extends for a fixed number of words.
- The generated poems demonstrated improved coherence with extended training.

Example Output

Input Seed:

"I wandered lonely as a"

Generated Poem (OneHotRNN):

‘‘I wandered lonely as a great had left behind a heart life shot, can laugh and free late? The pilot desert the child that by the women, sings of the country fast, And now we shall I’d To sing me all my only ride The ring beside the touch And now We with me beneath’’

Generated Poem (EmbeddingLSTM):

"I wandered lonely as a trifle, they are not the small returns to the inner were ever the to that sun, I can do nothing and my red them. And what are you? this all that is not by the great tomb of man. The golden sun, The planets, all the infinite host of heaven,"

COMPARISON AND ANALYSIS

Training Time & Loss:

- The **Embedding LSTM model** trained significantly faster (approximately **2x faster**) than the **One-Hot RNN model**.
- The **Embedding LSTM** achieved a substantially lower loss and improved practical efficiency.
- Despite the One-Hot RNN model achieving a slightly lower final loss, the **LSTM model** demonstrated more practical efficiency.

Text Quality:

- The **LSTM model** generated more meaningful, coherent, and structured poems.
- The **One-Hot RNN** model produced fragmented and less fluent text.

Advantages & Disadvantages:

- One-Hot Encoding RNN Model:
  - ➕ Simple to implement.
  - ➖ Slower training due to high-dimensional one-hot vectors.
  - ➖ Generates less coherent text.

- LSTM Model with Embeddings:
  - ➕ Faster training and improved accuracy.
  - ➕ Generates more creative and meaningful text.
  - ➖ Requires more memory due to trainable embeddings.

OBSERVATIONS AND CONCLUSIONS

- The **LSTM Model with Embeddings** demonstrated superior performance in terms of training efficiency and poem quality.
- The **One-Hot RNN Model** required significantly more training time and achieved a higher final loss.
- Future improvements such as **attention mechanisms**, **transformer models** or **hyperparameter tuning** may further enhance text quality.