Bangla Text Prediction Using Recurrent Neural Network (RNN)

Objective

Build a Bangla text prediction model capable of predicting the next word given a seed text. The model uses RNN to learn patterns from Bangla sentences.

Introduction

This project focuses on building a Bangla text prediction model using a Recurrent Neural Network (RNN). The goal is to predict the next word in a sentence based on a given seed text. By processing sequential patterns in Bangla paragraphs, the model learns the contextual relationships between words. Using an embedding layer and a SimpleRNN layer, the system can capture semantic and syntactic dependencies. This approach demonstrates how deep learning can be applied to natural language processing tasks in Bangla.

Data Preprocessing

Step 1: Load Paragraph

- Load the Bangla paragraph from a .txt file.
- Display a sample of the text to ensure correct loading.

Code:

```
file_path = "/content/drive/MyDrive/Colab Notebooks/MLL_7 - Copy.txt"

with open(file_path, "r", encoding="utf-8") as f:
    text = f.read()

print("✅ Paragraph Loaded!\n")
print("🔹 Sample Text (first 300 chars):\n", text[:300])
```

Step 2: Tokenization

Methodology:

- Use Tokenizer to convert words into unique integer indices.
- Calculate total unique words for embedding layer.

Code:

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text])
total_words = len(tokenizer.word_index) + 1

print(f"\n🔹 Total Unique Words: {total_words}")
print("🔹 Sample Word-Index Mapping (first 20):")
print(dict(list(tokenizer.word_index.items())[:20]))
```

Output:

```
🔹 Total Unique Words: 729
🔹 Sample Word-Index Mapping (first 20):
{'এবং': 1, 'মানুষের': 2, 'করে।': 3, 'ও': 4, 'প্রযুক্তি': 5,
```

Step 3: Create Sequences

Methodology:

- Convert tokenized text into n-gram sequences for training.
- Pad sequences to equal length to prepare features and labels.

Code:

```
token_list = tokenizer.texts_to_sequences([text])[0]
input_sequences = []

for i in range(1, len(token_list)):
    input_sequences.append(token_list[:i+1])

max_seq_len = max([len(seq) for seq in input_sequences])
input_sequences = pad_sequences(input_sequences, maxlen=max_seq_len, padding="pre")

X = input_sequences[:, :-1]
y = to_categorical(input_sequences[:, -1], num_classes=total_words)
```

Findings:

```
✅ Training Data Prepared
   🔹 X Shape: (1342, 1342), y Shape: (1342, 729)
```

- Features X and labels y prepared.
- Shapes of X and y indicate number of sequences and vocabulary size.

Step 4: Train-Test Split

Methodology:

- Split sequences into training and validation sets.
- 90% for training and 10% for validation.

Code:

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.1, random_state=42)
print(f"\n✅ Train-Test Split Done")
print(f"   🔹 Training Samples: {X_train.shape[0]}")
print(f"   🔹 Validation Samples: {X_val.shape[0]}")
```

Output:

```
🔹 Training Samples: 1207
🔹 Validation Samples: 135
```

Step 5: Build RNN Model

Methodology:

- Build a Sequential model with embedding, SimpleRNN, and Dense layers.
- Compile with categorical crossentropy and Adam optimizer.

Code:

```
model = Sequential([
    Embedding(input_dim=total_words, output_dim=128, input_length=max_seq_len-1),
    SimpleRNN(256),
    Dense(total_words, activation="softmax")
])

model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
```

Finding:

- A Sequential RNN model was successfully built with:
  - ✓ **Embedding layer**: converts word indices into 128-dimensional vectors.
  - ✓ **SimpleRNN layer**: 256 units to capture sequential dependencies.
  - ✓ **Dense output layer**: predicts probability for each word in the vocabulary.
- Ready for training with categorical crossentropy loss and Adam optimizer.

Step 6: Train Model

Methodology:

- Train the RNN model for 150 epochs with batch size 64.
- Monitor training and validation accuracy for performance.

Code:

```
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=150,
    batch_size=64,
    verbose=1
```

Step 7: Final Accuracy

Display final training and validation accuracy after all epochs.

```
🔹 Final Training Accuracy   : 63.38%
🔹 Final Validation Accuracy : 3.70%
```

Step 8: Prediction Function

Methodology:

- Define function predict_next_words(seed_text, next_words) to predict next word.
- Seed text is converted to sequences, padded, and fed to model.

Code:

```
def predict_next_words(seed_text, next_words=2):
    """Predict next words based on seed text"""
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list], maxlen=max_seq_len-1, padding="pre")
        predicted_index = np.argmax(model.predict(token_list, verbose=0), axis=-1)

        next_word = ""
        for word, idx in tokenizer.word_index.items():
            if idx == predicted_index:
                next_word = word
                break
        seed_text += " " + next_word
    return seed_text
```

Step 9: Test Predictions

Code:

```
print("Prediction 1:", predict_next_words("রোবোটিক সার্জারি", next_words=1))
print("Prediction 2:", predict_next_words("জীবনকে যুগান্তকারীভাবে", next_words=2))
print("Prediction 3:", predict_next_words("মানুষের আত্মার", next_words=3))
print("Prediction 4:", predict_next_words("উপাদানগুলোর", next_words=2))
print("Prediction 5:", predict_next_words("শীতে শীতল", next_words=2))
```

Output:

```
Prediction 1: রোবোটিক সার্জারি টেলিমেডিসিন
Prediction 2: জীবনকে যুগান্তকারীভাবে পরিবর্তন করেছে।
Prediction 3: মানুষের আত্মার পরিচয় বহন করে।
Prediction 4: উপাদানগুলোর সঙ্গে যুক্ত
Prediction 5: শীতে শীতল হাওয়া মানুষের
```

Original Text:

তেজস্বী সূর্য, বর্ষায় ঝরনার স্রোত এবং **শীতে শীতল হাওয়া মানুষের** অনুভূতি ও মনোবল
উপাদানগুলোর সঙ্গে যুক্ত; খাদ্য, জল, বস্ত্র, চিকিৎসা এবং শিক্ষার জন্য প্রকৃতি অপরিহ

প্রযুক্তি মানুষের **জীবনকে যুগান্তকারীভাবে পরিবর্তন করেছে**। কম্পিউটার, মোবাইল
দ্রুত, সহজ এবং কার্যকর করেছে। চিকিৎসা ক্ষেত্রে **রোবোটিক সার্জারি, টেলিমেডিসিন**
বৃদ্ধি করেছে। যোগাযোগ, বিনোদন, শিক্ষা, বিজ্ঞান এবং শিল্পে প্রযুক্তি মানুষের সৃজনশীলত

সংস্কৃতি **মানুষের আত্মার পরিচয় বহন করে**। ভাষা, সাহিত্য, কবিতা, নাটক, চিত্রকলা,

Discussion:

In this step, a Sequential RNN model was built for Bangla text prediction. The embedding layer converts word indices into dense 128-dimensional vectors, capturing semantic relationships. The SimpleRNN layer with 256 units processes these sequences, maintaining context from previous words to learn patterns in text. The dense output layer with softmax activation predicts the next word by assigning probabilities to all vocabulary words. The model summary shows the architecture and total trainable parameters. While this simple RNN can learn sequential dependencies effectively, it may struggle with long-range dependencies, which could be improved using LSTM or GRU layers in future iterations.