RNN Generative Text

Introduction:

In the world of language processing, the ability to predict the next character or word in a sequence holds significant importance, facilitating applications like text generation, spell checking, and autocomplete. **Recurrent neural networks (RNNs)** are super helpful for this, especially when it comes to guessing what the next letter might be. They've made a big difference in how well computers can handle language tasks like these.

Steps:

Character-Level Prediction:

1. **Data Collection and Preprocessing:**
   - Wikipedia content related to the provided title is fetched.
   - Text is cleaned by removing unnecessary elements such as links, emails, non-ASCII characters, and stop words. Text is also converted to lowercase and lemmatized.

2. **Model Training:**
   - Characters are tokenized.
   - Input-output pairs are generated from the tokenized sequences.
   - A character-level recurrent neural network (RNN) model is constructed, comprising an embedding layer, a SimpleRNN layer, and a dense output layer.
   - The model is trained on the generated input-output pairs.

3. **Prediction:**
   - Given input texts, the trained model predicts the next character in the sequence.

Word-Level Prediction:

1. **Data Collection and Preprocessing:**
   - Similar to the character-level prediction, Wikipedia content is fetched and cleaned.
   - This time, tokenization is performed at the word level.
2. **Model Training:**
   - Input-output pairs are generated from the tokenized word sequences.
   - A word-level RNN model is constructed with an embedding layer, a SimpleRNN layer, and a dense output layer.
   - The model is trained on input-output pairs.
3. **Prediction:**

- Given input texts, the trained model predicts the next word in the sequence.

Architecture 1: Character-Level Model

The character-level model utilizes a Simple Recurrent Neural Network (RNN) for character prediction. The architecture is structured as follows:

1. **Embedding Layer:**
   - Input Dimension: Vocabulary size.
   - Output Dimension: Embedding dimension.
   - Input Length: Maximum sequence length.
   - Description: Transforms input characters into dense vectors of fixed size.
2. **SimpleRNN Layer:**
   - Units: Number of RNN units
   - Description: Processes the embedded input sequences to capture sequential patterns.
3. **Dense Layer:**
   - Units: Vocabulary size
   - Activation: SoftMax
   - Description: Predicts the next character based on the output of the SimpleRNN layer.

Architecture 2: Word-Level Model

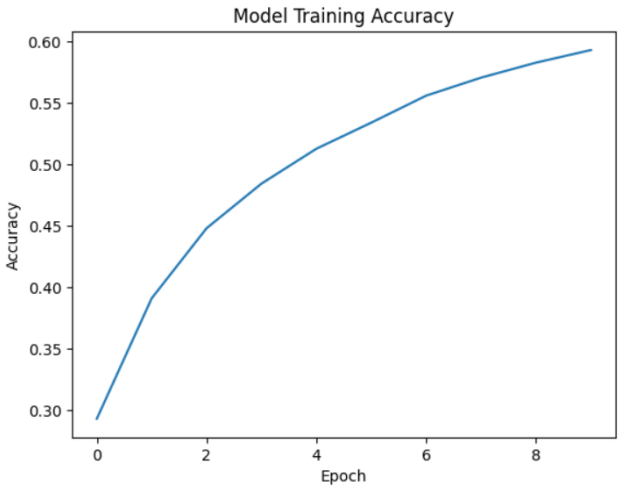
The word-level model also employs a Simple Recurrent Neural Network (RNN) for word prediction. The architecture is outlined as follows:

1. **Embedding Layer:**
   - Input Dimension: Vocabulary size.
   - Output Dimension: Embedding dimension.
   - Input Length: Maximum sequence length.
   - Description: Converts input words into dense vectors of fixed size.
2. **SimpleRNN Layer:**
   - Units: Number of RNN units
   - Description: Processes the embedded input sequences to capture sequential patterns.
3. **Dense Layer:**
   - Units: Vocabulary size
   - Activation: SoftMax
   - Description: Predicts the next word based on the output of the SimpleRNN layer.

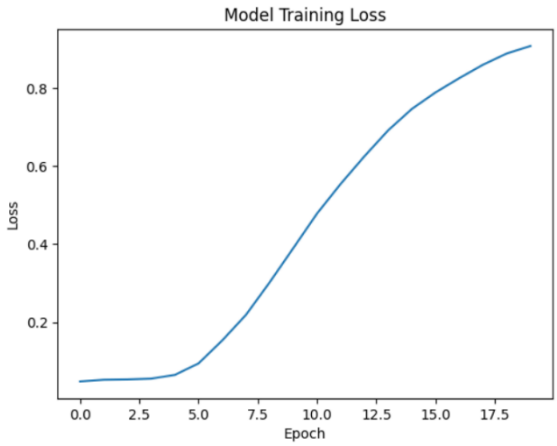
Comparison

Accuracy:

1. **Character-Based Model:**
   - Final Accuracy: **60.00%**
   - Loss: 1.3911



2. **Word-Based Model:**
   - Final Accuracy: **91.23%**
   - Loss: 0.9304



Commentary:

- **Accuracy**: The word-based model significantly outperforms the character-based model in terms of accuracy. The word-based model achieves an accuracy of 91.23%, indicating its capability to predict the next word in a sequence more accurately compared to the character-based model's accuracy of 59.09%.
- **Loss**: Additionally, the word-based model achieves a lower loss value (0.9304) compared to the character-based model's loss (1.3911). A lower loss value suggests that the word-based model is better at minimizing prediction errors during training, leading to more accurate predictions.

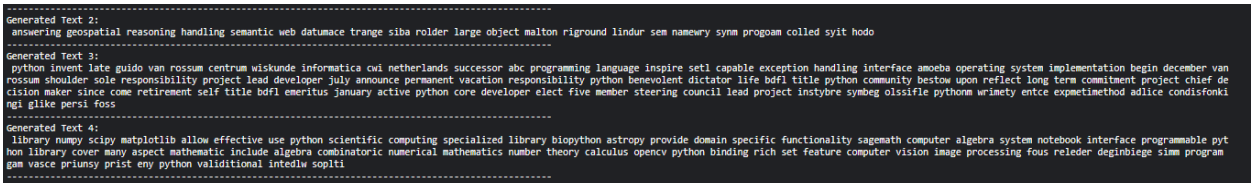
Test Char Model:

```
Input Text: python high
Predicted Next Word: level

Input Text: python programming
Predicted Next Word: language

Input Text: unofficial python implementation
Predicted Next Word: include
```

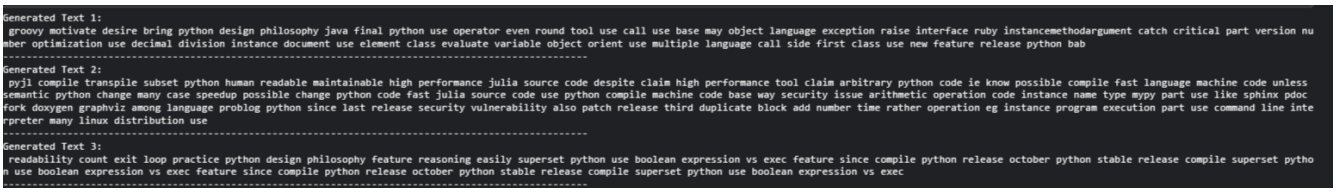
Test Word model:

```
Input Text: pytho
Predicted Next Word: n

Input Text: programmin
Predicted Next Word: g

Input Text: languag
Predicted Next Word: e

Input Text: cod
Predicted Next Word: e
```

Test Sequence of characters:



Test Sequence of Words:



Model 2: Enhanced Model

The enhanced model, denoted as Model 2, incorporates additional complexity compared to the initial model. This model employs a stack of Simple Recurrent Neural Network (RNN) layers to capture deeper hierarchical representations of the input data.

1. **Embedding Layer:**
   - Input Dimension: Vocabulary size.
   - Output Dimension: Embedding dimension.
   - Input Length: Maximum sequence length.
   - Description: Transforms input characters into dense vectors of fixed size.
2. **SimpleRNN Layers (Stacked):**
   - Units: Number of RNN units
   - Return Sequences: True (for all except the last layer)
   - Description: Sequentially processes the embedded input sequences, capturing increasingly abstract representations of the input data.
3. **Dense Layer:**
   - Units: Vocabulary size
   - Activation: SoftMax
   - Description: Predicts the next character based on the output of the final SimpleRNN layer.
4. **Training Dynamics:**
   - Epochs: The model is trained over **20 epochs.**
   - Accuracy: The model achieves a final accuracy of approximately **93.94%.**
   - Loss: The loss decreases over epochs, reaching a final value of approximately **0.5819.**

Conclusion:

In this project, we delved into both **character-level and word-level prediction** tasks using recurrent neural networks (RNNs) applied to Wikipedia content related to the topic of "(Python)". The objective was to predict the next character or word in a sequence given a partial input.

For **the character-level prediction task**, we implemented a character-level RNN model, trained it on the provided Wikipedia text, and evaluated its performance by predicting the next character for sample input sequences. The model demonstrated moderate accuracy in predicting the next characters, as evidenced by the generated predictions.

Similarly, we also explored **word-level prediction** by training another RNN model using word-level tokenization. This model was trained on the same Wikipedia text, predicting the next word in a sequence. The word-level model exhibited superior accuracy compared to the character-level model, highlighting the effectiveness of word-level representations in capturing linguistic patterns.