Recurrent Neural Network (RNN) for Text Generation

A Recurrent Neural Network (RNN) is a sequence-based neural network designed to process sequential data such as text. Unlike N-gram models, RNNs maintain a hidden state that carries information from previous time steps, allowing the model to capture context across an entire sequence. In text generation, an RNN learns patterns in text data and predicts the next character or word based on both the current input and past context, resulting in more coherent text generation.

```
# =========================
# RNN BASED TEXT GENERATION
# =========================

import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, Embedding

# -------------------------
# DATASET (SAME AS N-GRAM)
text = """
artificial intelligence is transforming modern society.
it is used in healthcare finance education and transportation.
machine learning allows systems to improve automatically with experience.
data plays a critical role in training intelligent systems.
large datasets help models learn complex patterns.
deep learning uses multi layer neural networks.
neural networks are inspired by biological neurons.
each neuron processes input and produces an output.
training a neural network requires optimization techniques.
gradient descent minimizes the loss function.

natural language processing helps computers understand human language.
text generation is a key task in nlp.
language models predict the next word or character.
recurrent neural networks handle sequential data.
lstm and gru models address long term dependency problems.
however rnn based models are slow for long sequences.

transformer models changed the field of nlp.
they rely on self attention mechanisms.
attention allows the model to focus on relevant context.
transformers process data in parallel.
this makes training faster and more efficient.
modern language models are based on transformers.

education is being improved using artificial intelligence.
intelligent tutoring systems personalize learning.
automated grading saves time for teachers.
online education platforms use recommendation systems.
technology enhances the quality of learning experiences.

ethical considerations are important in artificial intelligence.
fairness transparency and accountability must be ensured.
ai systems should be designed responsibly.
data privacy and security are major concerns.
researchers continue to improve ai safety.

text generation models can create stories poems and articles.
they are used in chatbots virtual assistants and content creation.
generated text should be meaningful and coherent.
evaluation of text generation is challenging.
human judgement is often required.

continuous learning is essential in the field of ai.
research and innovation drive technological progress.
students should build strong foundations in mathematics.
programming skills are important for ai engineers.
practical experimentation enhances understanding.
"""


text = text.lower().replace("\n", " ")

# -------------------------
# CHARACTER TOKENIZATION
# -------------------------
chars = sorted(list(set(text)))
char_to_index = {c: i for i, c in enumerate(chars)}
index_to_char = {i: c for c, i in char_to_index.items()}
```

```
index_to_char = {i: c for c, i in char_to_index.items()}

vocab_size = len(chars)
seq_length = 40

# -------------------------
# CREATE INPUT-OUTPUT SEQUENCES
# -------------------------
X = []
y = []

for i in range(len(text) - seq_length):
    X.append([char_to_index[c] for c in text[i:i+seq_length]])
    y.append(char_to_index[text[i+seq_length]])

X = np.array(X)
y = np.array(y)

# -------------------------
# BUILD RNN MODEL
# -------------------------
model = Sequential([
    Embedding(vocab_size, 64),
    SimpleRNN(128),
    Dense(vocab_size, activation='softmax')
])

model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer='adam'
)

# -------------------------
# TRAIN MODEL
# -------------------------
model.fit(X, y, epochs=10, batch_size=64)

# -------------------------
# TEXT GENERATION FUNCTION
# -------------------------
def generate_text(seed_text, length=300):
    output = seed_text
    for _ in range(length):
        seq = np.array([[char_to_index[c] for c in output[-seq_length:]]])
        prediction = model.predict(seq, verbose=0)
        next_char = index_to_char[np.argmax(prediction)]
        output += next_char
    return output

# -------------------------
# GENERATE TEXT
# -------------------------
seed = "artificial intelligence "
generated_text = generate_text(seed)

print("Generated Text using RNN:\n")
print(generated_text)
```

```
Epoch 1/10
34/34 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 3.1516
Epoch 2/10
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.8989
Epoch 3/10
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.8266
Epoch 4/10
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.6739
Epoch 5/10
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 2.5613
Epoch 6/10
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 2.4129
Epoch 7/10
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 2.2907
Epoch 8/10
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.1616
Epoch 9/10
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2.0553
Epoch 10/10
34/34 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1.9771
Generated Text using RNN:

artificial intelligence tion in protede te tecente tedeleng are tede te tecente tede tecente tede tecente te tean in purel t
```