Component-I: N Gram-Based Text Generation (Word Level)

```
import re
import random
from collections import defaultdict
```

```
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.
"""

text = text.lower()
text = re.sub(r'[^\w\s]', '', text)  # remove punctuation
words = text.split()
```

```
N = 3  # trigram model

ngram_model = defaultdict(list)

for i in range(len(words) - N + 1):
    context = tuple(words[i:i + N - 1])
    next_word = words[i + N - 1]
    ngram_model[context].append(next_word)
```

```
def generate_text_ngram(seed_text, num_words=25):
    seed_words = seed_text.lower().split()

    if len(seed_words) < N - 1:
        raise ValueError(f"Seed text must have at least {N-1} words")

    generated = seed_words.copy()

    for _ in range(num_words):
        context = tuple(generated[-(N - 1):])

        if context in ngram_model:
            next_word = random.choice(ngram_model[context])
            generated.append(next_word)
        else:
            break

    return " ".join(generated)
```

```
print(generate_text_ngram("artificial intelligence is transforming", 20))
print(generate_text_ngram("neural networks are inspired", 20))
```

```
artificial intelligence is transforming modern society it is used in healthcare finance education and transportation machine learning allows systems to improve automatically w
neural networks are inspired by biological neurons each neuron processes input and produces an output training a neural network requires optimization techniques gradient desce
```

Component-II: RNN-Based Text Generation (Word Level)

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
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.
"""

text = text.lower()
```

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text])

word_index = tokenizer.word_index
vocab_size = len(word_index) + 1

sequences = tokenizer.texts_to_sequences([text])[0]
```

```
SEQ_LEN = 5   # number of previous words

X, y = [], []

for i in range(SEQ_LEN, len(sequences)):
    X.append(sequences[i-SEQ_LEN:i])
    y.append(sequences[i])

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

```
model = Sequential([
    Embedding(vocab_size, 64, input_length=SEQ_LEN),
    SimpleRNN(128),
    Dense(vocab_size, activation="softmax")
])

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

model.summary()
```

```
Model: "sequential_17"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ embedding_17 (Embedding)        │ ?                      │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ simple_rnn_6 (SimpleRNN)        │ ?                      │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_27 (Dense)                │ ?                      │   0 (unbuilt) │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 0 (0.00 B)
 Trainable params: 0 (0.00 B)
 Non-trainable params: 0 (0.00 B)
```

```
model.fit(X, y, epochs=30, batch_size=32)
```

```
Epoch 1/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 5s 429ms/step - accuracy: 0.0000e+00 - loss: 4.1969
Epoch 2/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.1463 - loss: 4.1200
Epoch 3/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.3654 - loss: 4.0557
Epoch 4/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.4961 - loss: 3.9880
Epoch 5/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6275 - loss: 3.9238
Epoch 6/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6027 - loss: 3.8578
Epoch 7/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.6606 - loss: 3.7559
Epoch 8/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7100 - loss: 3.6558
Epoch 9/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.6567 - loss: 3.5422
Epoch 10/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6599 - loss: 3.4084
Epoch 11/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6151 - loss: 3.2543
Epoch 12/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6456 - loss: 3.0377
Epoch 13/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6788 - loss: 2.8543
Epoch 14/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6873 - loss: 2.6211
Epoch 15/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.6801 - loss: 2.3695
Epoch 16/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.7301 - loss: 2.1832
Epoch 17/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8843 - loss: 1.9233
Epoch 18/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.9448 - loss: 1.7362
Epoch 19/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.9558 - loss: 1.5082
Epoch 20/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.9701 - loss: 1.3378
Epoch 21/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.9890 - loss: 1.1394
Epoch 22/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.9811 - loss: 1.0044
Epoch 23/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 1.0000 - loss: 0.8355
Epoch 24/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 1.0000 - loss: 0.7514
Epoch 25/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 1.0000 - loss: 0.6024
Epoch 26/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 1.0000 - loss: 0.5210
Epoch 27/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 1.0000 - loss: 0.4570
Epoch 28/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 1.0000 - loss: 0.3997
Epoch 29/30
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 1.0000 - loss: 0.3536
```

```
def generate_text_rnn(seed_text, num_words=25):
    seed_text = seed_text.lower()

    for _ in range(num_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = token_list[-SEQ_LEN:]
        token_list = pad_sequences([token_list], maxlen=SEQ_LEN)

        predicted = np.argmax(model.predict(token_list, verbose=0))
        seed_text += " " + tokenizer.index_word.get(predicted, "")

    return seed_text
```

```
print(generate_text_rnn("artificial intelligence is"))
print(generate_text_rnn("neural networks are"))
```

```
artificial intelligence is in training society it is datasets help healthcare finance is patterns deep learning finance multi layer neural networks neural networks are inspire
neural networks are networks are biological neurons each neurons each input and networks and networks training a by a neurons optimization techniques processes techniques and
```

Component-III: LSTM-Based Text Generation (Word Level)

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```
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()
```

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text])

word_index = tokenizer.word_index
vocab_size = len(word_index) + 1

sequences = tokenizer.texts_to_sequences([text])[0]
```

```
SEQ_LEN = 5   # number of previous words used for prediction

X, y = [], []

for i in range(SEQ_LEN, len(sequences)):
    X.append(sequences[i-SEQ_LEN:i])
    y.append(sequences[i])

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

```
model = Sequential([
    Embedding(vocab_size, 64, input_length=SEQ_LEN),
    LSTM(128),
    Dense(vocab_size, activation="softmax")
])

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

model.summary()
```

**Model: "sequential_18"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_18 (Embedding) | ? | 0 (unbuilt) |
| lstm_6 (LSTM) | ? | 0 (unbuilt) |
| dense_28 (Dense) | ? | 0 (unbuilt) |

```
Total params: 0 (0.00 B)
Trainable params: 0 (0.00 B)
Non-trainable params: 0 (0.00 B)
```

```
model.fit(X, y, epochs=30, batch_size=32)
```

```
Epoch 1/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.0057 - loss: 5.2731
Epoch 2/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0935 - loss: 5.2613
Epoch 3/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.1153 - loss: 5.2481
Epoch 4/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0807 - loss: 5.2329
Epoch 5/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0432 - loss: 5.1901
Epoch 6/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0406 - loss: 5.0864
Epoch 7/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.0403 - loss: 4.9063
Epoch 8/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0452 - loss: 4.8302
Epoch 9/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.0723 - loss: 4.7586
Epoch 10/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0651 - loss: 4.6177
Epoch 11/30
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.1295 - loss: 4.4528
Epoch 12/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.1309 - loss: 4.2754
Epoch 13/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.1582 - loss: 4.0668
Epoch 14/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.1449 - loss: 3.8437
Epoch 15/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.1596 - loss: 3.6364
Epoch 16/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.1992 - loss: 3.3657
Epoch 17/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.2538 - loss: 3.1236
Epoch 18/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.2652 - loss: 2.9435
Epoch 19/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.3122 - loss: 2.7555
Epoch 20/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.4033 - loss: 2.4644
Epoch 21/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.4596 - loss: 2.2366
Epoch 22/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.5450 - loss: 2.0874
Epoch 23/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.6204 - loss: 1.8392
Epoch 24/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.6886 - loss: 1.5896
Epoch 25/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7845 - loss: 1.4504
Epoch 26/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7966 - loss: 1.3440
Epoch 27/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8426 - loss: 1.1637
Epoch 28/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8581 - loss: 1.0790
Epoch 29/30
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9123 - loss: 0.9485
```

```python
def generate_text_lstm(seed_text, num_words=25):
    seed_text = seed_text.lower()

    for _ in range(num_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = token_list[-SEQ_LEN:]
        token_list = pad_sequences([token_list], maxlen=SEQ_LEN)

        predicted = np.argmax(model.predict(token_list, verbose=0))
        seed_text += " " + tokenizer.index_word.get(predicted, "")

    return seed_text
```

```python
print(generate_text_lstm("artificial intelligence is"))
print(generate_text_lstm("neural networks are"))
```

```
artificial intelligence is and and society text in text a a understand neural techniques neural language is gradient transformers saves is is based is in is is is
neural networks are inspired biological biological neurons each neuron processes input and produces an output training a neural network requires optimization techniques gradie
```

Component-IV: Transformer-Based Text Generation (Word Level)

```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import (
    Embedding, Dense, LayerNormalization,
    Dropout, GlobalAveragePooling1D
)
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

```python
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()
```

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text])

sequences = tokenizer.texts_to_sequences([text])[0]
vocab_size = len(tokenizer.word_index) + 1
```

```
SEQ_LEN = 8  # number of previous words

X, y = [], []
for i in range(SEQ_LEN, len(sequences)):
    X.append(sequences[i-SEQ_LEN:i])
    y.append(sequences[i])

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

```
def positional_encoding(seq_len, d_model):
    positions = np.arange(seq_len)[:, np.newaxis]
    depths = np.arange(d_model)[np.newaxis, :] / d_model
    angle_rates = 1 / (10000 ** depths)
    angle_rads = positions * angle_rates
    return tf.cast(angle_rads, tf.float32)
```

```
class TransformerBlock(tf.keras.layers.Layer):
    def __init__(self, embed_dim, num_heads, ff_dim):
        super().__init__()
        self.att = tf.keras.layers.MultiHeadAttention(
            num_heads=num_heads, key_dim=embed_dim
        )
        self.ffn = tf.keras.Sequential([
            Dense(ff_dim, activation="relu"),
            Dense(embed_dim)
        ])
        self.norm1 = LayerNormalization()
        self.norm2 = LayerNormalization()
        self.drop1 = Dropout(0.1)
        self.drop2 = Dropout(0.1)

    def call(self, inputs, training=False):
        attn_output = self.att(inputs, inputs)
        attn_output = self.drop1(attn_output, training=training)
        out1 = self.norm1(inputs + attn_output)

        ffn_output = self.ffn(out1)
        ffn_output = self.drop2(ffn_output, training=training)
        return self.norm2(out1 + ffn_output)
```

```
EMBED_DIM = 32
NUM_HEADS = 1
FF_DIM = 64


inputs = tf.keras.Input(shape=(SEQ_LEN,))
embedding = Embedding(vocab_size, EMBED_DIM)(inputs)

pos_encoding = positional_encoding(SEQ_LEN, EMBED_DIM)
x = embedding + pos_encoding

x = TransformerBlock(EMBED_DIM, NUM_HEADS, FF_DIM)(x)
x = GlobalAveragePooling1D()(x)
outputs = Dense(vocab_size, activation="softmax")(x)

model = tf.keras.Model(inputs, outputs)

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

model.summary()
```

**Model: "functional_25"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_24 (InputLayer) | (None, 8) | 0 |
| embedding_19 (Embedding) | (None, 8, 32) | 6,240 |
| add_5 (Add) | (None, 8, 32) | 0 |
| transformer_block_5 (TransformerBlock) | (None, 8, 32) | 8,544 |
| global_average_pooling1d_5 (GlobalAveragePooling1D) | (None, 32) | 0 |
| dense_31 (Dense) | (None, 195) | 6,435 |

```
 Total params: 21,219 (82.89 KB)
 Trainable params: 21,219 (82.89 KB)
 Non-trainable params: 0 (0.00 B)
```

```
model.fit(X, y, epochs=50, batch_size=32)
```

```
Epoch 1/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 8s 252ms/step - accuracy: 0.0216 - loss: 5.3653
Epoch 2/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0322 - loss: 5.2908
Epoch 3/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0324 - loss: 5.2129
Epoch 4/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0268 - loss: 5.1404
Epoch 5/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0493 - loss: 5.0757
Epoch 6/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0264 - loss: 5.1102
Epoch 7/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0490 - loss: 5.0446
Epoch 8/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0191 - loss: 5.0476
Epoch 9/50
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0389 - loss: 5.0494
Epoch 10/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0312 - loss: 5.0418
Epoch 11/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0386 - loss: 5.0364
Epoch 12/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0240 - loss: 5.0211
Epoch 13/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0344 - loss: 5.0206
Epoch 14/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0378 - loss: 4.9283
Epoch 15/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0455 - loss: 4.9094
Epoch 16/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0348 - loss: 4.9173
Epoch 17/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0431 - loss: 4.8884
Epoch 18/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0483 - loss: 4.7971
Epoch 19/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0421 - loss: 4.7754
Epoch 20/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0692 - loss: 4.7373
Epoch 21/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0957 - loss: 4.7683
Epoch 22/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.0802 - loss: 4.7063
Epoch 23/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0864 - loss: 4.6876
Epoch 24/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0804 - loss: 4.6328
Epoch 25/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0924 - loss: 4.5745
Epoch 26/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.1486 - loss: 4.4643
Epoch 27/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.1404 - loss: 4.3687
Epoch 28/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.2060 - loss: 4.1382
Epoch 29/50
```

```
def generate_text_transformer(seed_text, num_words=25):
    seed_text = seed_text.lower()

    for _ in range(num_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = token_list[-SEQ_LEN:]
        token_list = pad_sequences([token_list], maxlen=SEQ_LEN)

        predicted = np.argmax(model.predict(token_list, verbose=0))
        seed_text += " " + tokenizer.index_word.get(predicted, "")

    return seed_text
```

```
print(generate_text_transformer("artificial intelligence is"))
print(generate_text_transformer("neural networks are"))
```

```
artificial intelligence is data data data ai systems used systems systems systems systems systems and deep learning deep deep deep learning learning neural networks are networ
neural networks are data the the the the the processes creation dependency understand understand understand understand understand neural evaluation for task task task task is
```