Generative AI Assignment

Text Generation Using LSTM Neural Networks

Student: Adetola Ridwat Odulaja
Course: Data Analytics & Artificial Intelligence
Date: November 2025

Introduction

Generative Artificial Intelligence (Generative AI) refers to a class of machine learning techniques that enable machines to produce new content such as text, images, audio, and code. These systems learn patterns from large datasets and generate creative, human-like outputs.
In recent years, transformer-based models such as GPT (Generative Pre-trained Transformers) have revolutionized language generation by using attention mechanisms and large-scale training.

This assignment explores the fundamentals of Generative AI, focusing on its significance, architecture, use cases, and ethical considerations. It also includes a practical implementation of a text generation model trained on a public domain dataset.

GPT Architecture and Functionality

GPT models are built on the Transformer Decoder architecture. Their key components include:

• Self-Attention

Allows the model to determine which words in a sentence are most relevant when predicting the next token.

• Multi-Head Attention

Multiple attention heads run in parallel to capture different linguistic relationships such as grammar and meaning.

• Positional Encoding

Provides the model with information about the order of words since transformers do not process data sequentially by default.

• Feedforward Neural Networks

Applies non-linear transformations to improve pattern recognition.

• Autoregressive Text Generation

GPT generates text one token at a time using probability distributions learned during training.

Together, these mechanisms allow GPT models to generate fluent, contextual, and human-like text.

Dataset Preparation

For this project, I used "Alice in Wonderland" by Lewis Carroll, obtained from Project Gutenberg (public domain text).
The text was cleaned, converted to lowercase, tokenized, and transformed into input sequences for training the model.

A word-level Tokenizer was used to reduce memory usage and ensure the training process runs efficiently within Google Colab.

```
from google.colab import files
uploaded = files.upload()

file_name = next(iter(uploaded))
text = open(file_name, 'r', encoding='utf-8').read().lower()
print("Loaded:", file_name)
```

```
Choose Files alice.txt.crdownload
alice.txt.crdownload(n/a) - 174355 bytes, last modified: 11/15/2025 - 100% done
Saving alice.txt.crdownload to alice.txt (1).crdownload
Loaded: alice.txt (1).crdownload
```

Clean and Tokenize

```
import re
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Clean text
text = re.sub(r'[^a-zA-Z0-9\s]', ' ', text)

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

print("Total words:", total_words)
```

```
Total words: 3031
```

```
input_sequences = []
words = text.split()

seq_length = 20  # small size for RAM efficiency

for i in range(seq_length, len(words)):
    seq = words[i-seq_length:i+1]
    encoded = tokenizer.texts_to_sequences([' '.join(seq)])[0]
    input_sequences.append(encoded)

input_sequences = np.array(input_sequences)
print("Total sequences:", input_sequences.shape)
```

```
Total sequences: (30559, 21)
```

```
X = input_sequences[:, :-1]
y = input_sequences[:, -1]

y = tf.keras.utils.to_categorical(y, num_classes=total_words)
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(total_words, 128, input_length=seq_length),
    tf.keras.layers.LSTM(128),
    tf.keras.layers.Dense(total_words, activation="softmax")
])

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

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecate
  warnings.warn(
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ embedding (Embedding)           │ ?                      │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm (LSTM)                     │ ?                      │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ ?                      │   0 (unbuilt) │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 0 (0.00 B)
 Trainable params: 0 (0.00 B)
```

```
model.fit(X, y, epochs=5, batch_size=128, verbose=1)
```

```
Epoch 1/5
239/239 ━━━━━━━━━━━━━━━━━━━━ 24s 87ms/step - accuracy: 0.0455 - loss: 6.7801
```

```
Epoch 2/5
239/239 ━━━━━━━━━━━━━━━━━━━━ 23s 97ms/step - accuracy: 0.0607 - loss: 5.9615
Epoch 3/5
239/239 ━━━━━━━━━━━━━━━━━━━━ 21s 88ms/step - accuracy: 0.0738 - loss: 5.7716
Epoch 4/5
239/239 ━━━━━━━━━━━━━━━━━━━━ 21s 88ms/step - accuracy: 0.0929 - loss: 5.5827
Epoch 5/5
239/239 ━━━━━━━━━━━━━━━━━━━━ 21s 87ms/step - accuracy: 0.1137 - loss: 5.3867
<keras.src.callbacks.history.History at 0x7d4c73f12f60>
```

```python
def generate_text(seed_text, next_words=30):
    for _ in range(next_words):
        encoded = tokenizer.texts_to_sequences([seed_text])[0]
        encoded = pad_sequences([encoded], maxlen=seq_length, truncating='pre')

        pred = model.predict(encoded, verbose=0)
        next_word = tokenizer.index_word[np.argmax(pred)]
        seed_text += " " + next_word
    return seed_text

print(generate_text("alice looked around", 40))
```

```
alice looked around the queen and the queen and the queen and the queen and the queen and the queen and the queen and the queen
```

Results

The LSTM model was trained for 5 epochs with a sequence length of 20 words.
During training, accuracy improved steadily from approximately **4%** to **11%**, showing that the model successfully learned basic next-word prediction patterns.

A sample generated output from the seed text “alice looked around” produced repetitive but grammatically connected text. This behavior is typical of small LSTM models, especially when trained on limited data without advanced attention mechanisms.

Overall, the model demonstrated the core principles of generative modeling—learning distributions and predicting the next word based on context.

Ethical Considerations in Generative AI

Generative AI raises several important ethical issues, including:

• Bias

Models may learn harmful stereotypes if trained on biased data.

• Misinformation

AI systems may generate information that appears real but is factually incorrect.

• Privacy

Models trained on unfiltered datasets may unintentionally reproduce private or sensitive information.

• Copyright Concerns

Using copyrighted training data without permissions can cause legal and ethical risks.

• Responsible Use

Generative AI should not be deployed in ways that cause harm, deceive users, or manipulate public opinion.

Mitigating these risks requires transparent data sourcing, bias audits, human monitoring, and responsible deployment practices.

Conclusion

In this assignment, I explored the foundations of Generative AI, focusing on transformer-based models like GPT and how they generate human-like text. The hands-on component involved training a lightweight LSTM text generation model on the public domain text “Alice in Wonderland,” covering the end-to-end workflow from data cleaning and tokenization to model training and inference.

Summary of Results

The model successfully learned basic language patterns and demonstrated next-word prediction capabilities. Training accuracy improved from approximately 4% to 11%, and the model generated continuous text given a starting prompt. Although the output contained repetition and lacked deeper coherence compared to advanced GPT models, it illustrated the essential mechanisms behind generative modeling.

This project also highlighted key ethical considerations related to bias, misinformation, privacy, and responsible AI deployment.

Overall, this assignment deepened my understanding of both theoretical and practical aspects of Generative AI. Future work may include experimenting with transformer-based architectures, fine-tuning pretrained GPT models, or exploring multimodal generative systems that combine text with images, audio, and other data types.

References

• Project Gutenberg – https://www.gutenberg.org
• TensorFlow Documentation – https://www.tensorflow.org
• Keras Tokenizer API – https://keras.io/api/preprocessing/text
• “Attention Is All You Need” (Vaswani et al., 2017)