Assignment Code: DS-AG-031

Generative AI - Text Generation and Machine Translation | **Assignment**

Instructions: Carefully read each question. Use Google Docs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

Total Marks: 200

Question 1: What is Generative AI and what are its primary use cases across industries?

Answer:

Generative AI refers to a class of artificial intelligence models that can generate new content such as text, images, audio, video, or code based on patterns learned from existing data. Unlike traditional AI systems that classify or predict, generative models create new data samples that resemble the training data.

Generative AI typically uses deep learning architectures such as Transformers, Variational Autoencoders (VAEs), and Generative Adversarial Networks (GANs).

Primary Use Cases Across Industries:

1. **Healthcare**
    - Drug discovery
    - Medical report generation
    - Synthetic medical data creation
2. **Education**
    - Automated tutoring systems
    - Question generation
    - Content summarization
3. **Media & Entertainment**
    - Script writing
    - Music and poetry generation
    - Image and video generation
4. **Finance**

- Report generation
- Fraud pattern simulation
- Risk modeling

5. **E-commerce**
   - Product description generation
   - Personalized marketing content
   - Chatbots
6. **Software Development**
   - Code generation
   - Debugging assistance
   - Documentation writing

Question 2: Explain the role of probabilistic modeling in generative models. How do these models differ from discriminative models?

Answer:

Role of Probabilistic Modeling

Generative models learn the joint probability distribution:

$$P(X, Y)$$

or simply:

$$P(X)$$

They model how data is generated by estimating the probability distribution of the input data. For example, in text generation, the model predicts:

$P(word_t|word_1,word_2,...,word_{t-1})$

Difference Between Generative and Discriminative Models

Feature	**Generative Model**	**Discriminative Model**
Learns	Joint probability P(X,Y)	Conditional probability P(Y
Purpose	Generate data	Classify data
Example	GPT, VAE, GAN	Logistic Regression, SVM
Output	New samples	Labels

Question 3: What is the difference between Autoencoders and Variational Autoencoders (VAEs) in the context of text generation?

Answer:

Autoencoder

- Deterministic model
- Compresses input into latent vector
- Reconstructs original input
- Used mainly for dimensionality reduction

Architecture:
Input → Encoder → Latent Vector → Decoder → Output

Variational Autoencoder (VAE)

- Probabilistic model
- Encodes input as a distribution (mean and variance)
- Uses sampling via reparameterization trick
- Enables generation of new data

Architecture:
Input → Encoder → (μ, σ) → Sampling → Decoder → Output

Question 4: Describe the working of attention mechanisms in Neural Machine Translation (NMT). Why are they critical?

Answer:

In Neural Machine Translation (NMT), attention allows the decoder to focus on relevant words in the input sentence while generating each output word.

Working:

1. Encoder processes input sentence → produces hidden states.
2. Decoder generates word-by-word.
3. Attention calculates alignment scores between decoder state and encoder states.
4. Weighted sum (context vector) is computed.
5. Context vector helps generate accurate output word.

Why Critical?

- Solves long sentence problem
- Improves translation quality
- Captures contextual relationships
- Enables transformer architecture

Question 5: What ethical considerations must be addressed when using generative AI for creative content such as poetry or storytelling?

Answer:

1. **Bias and Fairness**
   - Models may reflect societal biases.
2. **Plagiarism**
   - Risk of copying existing content.
3. **Misinformation**
   - Fake news generation.
4. **Intellectual Property**
   - Ownership of AI-generated content.
5. **Deepfakes**
   - Identity misuse.
6. **Accountability**
   - Who is responsible for harmful outputs?

Question 6: Use the following small text dataset to train a simple Variational Autoencoder (VAE) for text reconstruction:

["The sky is blue", "The sun is bright", "The grass is green",
"The night is dark", "The stars are shining"]

1. Preprocess the data (tokenize and pad the sequences).
2. Build a basic VAE model for text reconstruction.
3. Train the model and show how it reconstructs or generates similar sentences.

Include your code, explanation, and sample outputs.

(*Include your Python code and output in the code box below.*)
Answer:

Code:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Input, Dense, Lambda
from tensorflow.keras.models import Model
from tensorflow.keras import backend as K

# Dataset
sentences = [
    "The sky is blue",
    "The sun is bright",
    "The grass is green",
    "The night is dark",
    "The stars are shining"
]

# Tokenization
tokenizer = Tokenizer()
tokenizer.fit_on_texts(sentences)
sequences = tokenizer.texts_to_sequences(sentences)

max_len = max(len(seq) for seq in sequences)
padded = pad_sequences(sequences, maxlen=max_len, padding='post')

vocab_size = len(tokenizer.word_index) + 1

# Simple VAE
input_dim = max_len
latent_dim = 2

inputs = Input(shape=(input_dim,))
h = Dense(16, activation='relu')(inputs)
z_mean = Dense(latent_dim)(h)
z_log_var = Dense(latent_dim)(h)
```

```
def sampling(args):
    z_mean, z_log_var = args
    epsilon = K.random_normal(shape=(K.shape(z_mean)[0], latent_dim))
    return z_mean + K.exp(0.5 * z_log_var) * epsilon

z = Lambda(sampling)([z_mean, z_log_var])

decoder_h = Dense(16, activation='relu')
decoder_output = Dense(input_dim, activation='sigmoid')

h_decoded = decoder_h(z)
outputs = decoder_output(h_decoded)

vae = Model(inputs, outputs)
vae.compile(optimizer='adam', loss='mse')

vae.fit(padded, padded, epochs=200, verbose=0)

reconstructed = vae.predict(padded)
print("Reconstructed Output:")
print(np.round(reconstructed))
```

Question 7: Use a pre-trained GPT model (like GPT-2 or GPT-3) to translate a short English paragraph into French and German. Provide the original and translated text.

(*Include your Python code and output in the code box below.*)

Answer:

```
 Code:
# Install if not installed
!pip install -q transformers sentencepiece sacremoses

from transformers import MarianMTModel, MarianTokenizer

# Define input text FIRST
text = "Artificial Intelligence is transforming the world."

# Translation function
def translate(text, model_name):
    tokenizer = MarianTokenizer.from_pretrained(model_name)
    model = MarianMTModel.from_pretrained(model_name)
    tokens = tokenizer(text, return_tensors="pt", padding=True)
    translated = model.generate(**tokens)
    return tokenizer.decode(translated[0], skip_special_tokens=True)

# Translate
print("Original:", text)
print("French:", translate(text, "Helsinki-NLP/opus-mt-en-fr"))
print("German:", translate(text, "Helsinki-NLP/opus-mt-en-de"))
```

Question 8: Implement a simple attention-based encoder-decoder model for English-to-Spanish translation using Tensorflow or PyTorch.

(*Include your Python code and output in the code box below.*)

Answer:

Code:

```
import torch
import torch.nn as nn

class Encoder(nn.Module):
    def __init__(self, input_dim, emb_dim, hid_dim):
        super().__init__()
        self.embedding = nn.Embedding(input_dim, emb_dim)
        self.rnn = nn.GRU(emb_dim, hid_dim)

    def forward(self, src):
        embedded = self.embedding(src)
        outputs, hidden = self.rnn(embedded)
        return outputs, hidden

class Attention(nn.Module):
    def __init__(self, hid_dim):
        super().__init__()
        self.attn = nn.Linear(hid_dim*2, hid_dim)
        self.v = nn.Linear(hid_dim, 1, bias=False)

    def forward(self, hidden, encoder_outputs):
        seq_len = encoder_outputs.shape[0]
        hidden = hidden.repeat(seq_len, 1, 1)
        energy = torch.tanh(self.attn(torch.cat((hidden, encoder_outputs), dim=2)))
        attention = self.v(energy).squeeze(2)
        return torch.softmax(attention, dim=0)
```

Question 9: Use the following short poetry dataset to simulate poem generation with a pre-trained GPT model:

["Roses are red, violets are blue,",
"Sugar is sweet, and so are you.",
"The moon glows bright in silent skies,",
"A bird sings where the soft wind sighs."]

Using this dataset as a reference for poetic structure and language, generate a new 2-4 line poem using a pre-trained GPT model (such as GPT-2). You may simulate fine-tuning by prompting the model with similar poetic patterns.

Include your code, the prompt used, and the generated poem in your answer.

(*Include your Python code and output in the code box below.*)
Answer:

```
 Code:
from transformers import pipeline

generator = pipeline("text-generation", model="gpt2")

prompt = "Roses are red, violets are blue,"
poem = generator(prompt, max_length=40, num_return_sequences=1)

print(poem[0]['generated_text'])
```

Question 10: Imagine you are building a creative writing assistant for a publishing company. The assistant should generate story plots and character descriptions using

Generative AI. Describe how you would design the system, including model selection, training data, bias mitigation, and evaluation methods. Explain the real-world challenges you might face.

(*Include your Python code and output in the code box below.*)

Answer:

System Design

1. **Model Selection**
    - Use GPT-based transformer model.
    - Fine-tune on novels and story datasets.
2. **Training Data**
    - Public domain books
    - Genre-specific corpora
    - Character description datasets
3. **Bias Mitigation**
    - Balanced datasets
    - Toxicity filtering
    - Human-in-the-loop review
4. **Evaluation Methods**
    - BLEU score
    - Human evaluation
    - Creativity scoring

5. Architecture

- Input: Genre + Theme
- Output: Plot + Characters
- Interface: Web-based editor

Challenges

- Bias in storytelling
- Copyright issues
- Maintaining originality
- Controlling hallucinations
- Ethical content moderation