K. J. Somaiya School of Engineering, Mumbai-77

Batch: B3 Roll No.:16010122257
Experiment / assignment / tutorial No:9

Title: Text Generation using GPT-2

Objectives:

- To introduce students to the concept of Generative AI.
- To understand how GPT-2 generates text using prompt-based inputs.
- To explore how different decoding strategies and hyper-parameters affect generated output.

Expected Outcome of Experiment:

Course Outcome	After successful completion of the course students should be able to
CO 4	Analyze applications of AI and understand planning & learning processes in advanced AI applications

---

Resources & References

1. HuggingFace GPT-2: https://huggingface.co/gpt2 , last retrieved on April 02,2025
2. Illustrated Transformer: https://jalammar.github.io/illustrated-transformer/ , last retrieved on April 02,2025
3. GPT-2 Paper: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf , last retrieved on April 02,2025
4. HuggingFace Blog on Generation: https://huggingface.co/blog/how-to-generate , last retrieved on April 02,2025
5. https://cloud.google.com/ai/generative-ai?hl=en , last retrieved on April 02,2025

Text generation is a natural language processing (NLP) task where a model creates meaningful and coherent text based on a given input or prompt. It is a core capability of Generative AI and is commonly achieved using deep learning models, especially transformer-based architectures like GPT (Generative Pre-trained Transformer). These models are trained on massive corpora of text data and learn language patterns, grammar, and context. In text generation, the model predicts the next word (or token) one step at a time, using the context of the previous words, and continues until it

reaches a desired length or stopping condition. Applications include story writing, code generation, chatbots, summarization, and creative content creation.

Generative AI refers to a branch of artificial intelligence that is capable of creating new content — such as text, images, music, or even code — based on patterns learned from large datasets. Unlike traditional AI that focuses on classification or prediction tasks, generative AI learns to produce original data that resembles its training input. In natural language processing (NLP), generative AI models like GPT can generate human-like text when given a prompt, making them useful for applications like chatbots, story writing, summarization, and more.

GPT-2 (Generative Pre-trained Transformer 2) is a powerful language model developed by OpenAI that uses deep learning to generate human-like text. It is based on the transformer architecture and was trained on a large corpus of internet text. Given a starting prompt, GPT-2 can continue writing coherent and contextually relevant sentences. It works by predicting the next word in a sequence using the context of previous words, making it capable of generating creative content, answering questions, translating text, and more.

In text generation, decoding strategies are methods used to determine which words the model should generate next. Common strategies include **Greedy Search** (picks the most probable word), **Beam Search** (keeps multiple best options), and Sampling methods like Top-k and Top-p that add randomness.

In the context of **text generation** using large language models like GPT-2, the process of "**hyper-parameter tuning**" refers to adjusting the generation parameters to control how the text is generated. These hyper-parameters influence the model's **creativity**, **coherence, repetition, and overall quality of the output**.

Common Hyper-parameters for Text Generation

Hyperparameter	Description	Typical Range / Values
max_length	Total number of tokens (words + punctuation) to generate.	20–100
temperature	Controls randomness. Lower = more conservative, Higher = more creative.	0.5 – 1.5 (default = 1.0)
top_k	Randomly samples from the top K most probable next words.	30–100
top_p (nucleus)	Chooses from the smallest set of tokens whose cumulative probability ≥ p.	0.8 – 0.95
do_sample	Enables sampling instead of greedy or beam decoding.	True / False
repetition_penalty	Penalizes repeated phrases. Higher = less repetition.	1.0 (no penalty), 1.2+
num_beams	Number of beams used in **Beam Search** to explore multiple paths.	1 (greedy), 3–10

early_stopping	Stops generation when all beams are finished. Used with beam search.	True / False
num_return_sequences	Number of output sequences to return for each input prompt.	1 – 5

How The Hyper-parameters Affect the Output:

Parameter	Effect on Output
temperature=1.5	More diverse and creative output
temperature=0.5	Safer and more repetitive output
top_k=50	Random, but constrained to top 50 choices
top_p=0.9	More dynamic; considers word distribution
repetition_penalty=1.2	Avoids looping or repetitive phrases
num_beams=5	Improves fluency and quality at the cost of speed

Examples:

Default (greedy):

model.generate(inputs, max_length=50)

High temperature (more creative):

model.generate(inputs, max_length=50, temperature=1.5)

Top-k sampling:

model.generate(inputs, max_length=50, do_sample=True, top_k=50)

Top-p (nucleus sampling):

model.generate(inputs, max_length=50, do_sample=True, top_p=0.9)

Beam search:

model.generate(inputs, max_length=50, num_beams=5, early_stopping=True)

Instructions for Students:

1. Open Google Colab and create a new notebook**.**
2. Copy and run the following base code to generate text using the GPT-2 model:

```
!pip install transformers torch

from transformers import GPT2LMHeadModel, GPT2Tokenizer

# Load model and tokenizer
model = GPT2LMHeadModel.from_pretrained("gpt2")
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

# Write your prompt
prompt = "Artificial Intelligence will revolutionize education because"
inputs = tokenizer.encode(prompt, return_tensors="pt")

# Generate text
outputs = model.generate(inputs, max_length=50, temperature=0.7,
num_return_sequences=1)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

---

3. Now**,** experiment by adding different hyper-parameters to the generate() function one by one and in combinations. Some hyper-parameters you must try:

   - temperature
   - top_k
   - top_p
   - num_beams
   - repetition_penalty
   - do_sample
   - max_length
   - num_return_sequences

4. Run at least 5 different configurations, each with a new combination of hyper-parameters.
5. For each configuration, record:

   - The generated output
   - A short observation (Was it more creative? repetitive? random? logical?)

```
[1] !pip install transformers torch
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 54.9 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 41.2 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.6 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 6.3 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 10.7 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.2 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.9 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 81.2 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
```

```
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
```

```
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-cu12-12.4.127 nvidia-cud
```

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer

model = GPT2LMHeadModel.from_pretrained("gpt2")
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

prompt = "Artificial Intelligence will revolutionize education because"
inputs = tokenizer.encode(prompt, return_tensors="pt")

outputs = model.generate(inputs, max_length=50, temperature=0.7, num_return_sequences=1)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab an
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100% 665/665 [00:00<00:00, 45.6kB/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For better performance, install the packa
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For
model.safetensors: 100% 548M/548M [00:06<00:00, 74.9MB/s]
```

```
generation_config.json: 100% 124/124 [00:00<00:00, 9.06kB/s]
tokenizer_config.json: 100% 26.0/26.0 [00:00<00:00, 1.91kB/s]
vocab.json: 100% 1.04M/1.04M [00:00<00:00, 15.6MB/s]
merges.txt: 100% 456k/456k [00:00<00:00, 8.36MB/s]
tokenizer.json: 100% 1.36M/1.36M [00:00<00:00, 36.5MB/s]
/usr/local/lib/python3.11/dist-packages/transformers/generation/configuration_utils.py:628: UserWarning: `do_sample` is set to `False`. However, `temperature` is
  warnings.warn(
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
The attention mask is not set and cannot be inferred from input because pad token is same as eos token. As a consequence, you may observe unexpected behavior. Pl
Artificial Intelligence will revolutionize education because it will be able to create a new generation of students who will be able to learn from the best teach

The new generation of students will be able to
```

Record your observations: (Add maximum outputs as you can add)

Configuration-1:

```
[4] from transformers import GPT2LMHeadModel, GPT2Tokenizer

model = GPT2LMHeadModel.from_pretrained("gpt2")
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

prompt = "Artificial Intelligence will revolutionize education because"
inputs = tokenizer.encode(prompt, return_tensors="pt")

outputs = model.generate(inputs, max_length=50, temperature=1.0, top_p=0.9, do_sample=True, num_return_sequences=1)

print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

- Parameters and their values: max_length=50, temperature=1.0, top_p=0.9, do_sample=True, num_return_sequences=1
- Output received:

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Artificial Intelligence will revolutionize education because the best way to do it is to get it right with AI. The problem with AI, according to me, is that it isn't the kind of system I want to develop. I'm a big believer
```

- Explanation/Learning: The output was logical and followed a coherent structure. The temperature setting of 0.7 maintained a balance between creativity and structure, producing readable and sensible results.

Configuration-2:

```
[8]
14s

from transformers import GPT2LMHeadModel, GPT2Tokenizer

model = GPT2LMHeadModel.from_pretrained("gpt2")
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

prompt = "Artificial Intelligence will revolutionize education because"
inputs = tokenizer.encode(prompt, return_tensors="pt")

outputs = model.generate(inputs, max_length=50, num_beams=5, repetition_penalty=1.2)

print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

- Parameters and their values: max_length=50, num_beams=5, repetition_penalty=1.2
- Output received:

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Artificial Intelligence will revolutionize education because it will make it possible for students to learn about the world around them.

This is not a new idea. In fact, it has been around for a long time. In the early 1990s,
```

- Explanation/Learning: Using **beam search** (`num_beams=5`) enhanced the logical structure and fluency of the output. The sentence was coherent, informative, and followed a clear narrative. The `repetition_penalty=1.2` helped reduce redundant phrases. Overall, this configuration delivered a well-constructed, fact-style response—ideal for educational or formal content generation.

Configuration-3:

```
[9]
from transformers import GPT2LMHeadModel, GPT2Tokenizer

model = GPT2LMHeadModel.from_pretrained("gpt2")
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

prompt = "Artificial Intelligence will revolutionize education because"
inputs = tokenizer.encode(prompt, return_tensors="pt")

outputs = model.generate(inputs, max_length=50, temperature=0.8, top_k=50, do_sample=True)

print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

- Parameters and their values: max_length=50, temperature=0.8, top_k=50, do_sample=True
- Output received:

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Artificial Intelligence will revolutionize education because it will be able to change the way we think, live and work.

"The machine has come to be the arbiter, the arbitrator, the deciding factor," says Dr. Ben Brant
```

- Explanation/Learning: This configuration uses **top-k sampling** with `do_sample=True` and a **moderate temperature** of 0.8, which introduces controlled creativity. The output is imaginative yet coherent, with a quote that adds a realistic touch. The inclusion of a fictional expert and metaphorical phrasing shows how top-k sampling can enrich narratives while staying on topic. This makes it suitable for more expressive, article-like outputs.

Configuration-4:

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer

model = GPT2LMHeadModel.from_pretrained("gpt2")
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

prompt = "Artificial Intelligence will revolutionize education because"
inputs = tokenizer.encode(prompt, return_tensors="pt")

outputs = model.generate(inputs, max_length=50, num_beams=10, repetition_penalty=1.5)

print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

- Parameters and their values:max_length=50, num_beams=10, repetition_penalty=1.5
- Output received:

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Artificial Intelligence will revolutionize education because it will transform the way we think about science, technology, engineering, and math. It will change the way we think about science, technology, engineering, and math. It will transform the way we think about
```

- Explanation/Learning: Despite a high beam width (`num_beams=10`) and an increased repetition penalty (`1.5`), the output became **overly repetitive**. The model seemed stuck in a loop, repeating phrases. This indicates that beam search, especially at high values, may reduce randomness but can sometimes cause loops if not paired with appropriate sampling or more aggressive penalties. It's a great example of how more deterministic decoding can backfire without balance.

Configuration-5:

```python
from transformers import GPT2LMHeadModel, GPT2Tokenizer

model = GPT2LMHeadModel.from_pretrained("gpt2")
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

prompt = "Artificial Intelligence will revolutionize education because"
inputs = tokenizer.encode(prompt, return_tensors="pt")

outputs = model.generate(
    inputs,
    max_length=50,
    do_sample=True,
    temperature=0.9,
    top_p=0.95,
    num_return_sequences=3
)

for i in range(3):
    print(f"Generated Output {i+1}:")
    print(tokenizer.decode(outputs[i], skip_special_tokens=True))
    print("\n")
```

- Parameters and their values: max_length=50, temperature=0.9, top_p=0.95, do_sample=True, num_return_sequences=3
- Output received:

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Generated Output 1:

Artificial Intelligence will revolutionize education because the problem with teaching artificial intelligence is that it's so much more complex and confusing than learning an actual language.

In fact, one big problem with artificial intelligence is that we have to figure out how to

Generated Output 2:

Artificial Intelligence will revolutionize education because it will give students access to a richer and more intuitive understanding of the world around them, which will lead them to better understand and practice cognitive skills.

The next step in AI, called artificial intelligence,

Generated Output 3:

Artificial Intelligence will revolutionize education because it will be a big part of the solution," said Mark Gavril, director of the Harvard-Smithsonian Center for Astrophysics's Advanced Micro Devices Research (AMRIAD) Project on AI.
```

- Explanation/Learning: The warning about missing `attention_mask` and `pad_token_id` suggests potential unreliable behavior; setting these helps the model process input more accurately, especially for padded sequences.

Configuration-6:

```python
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("gpt2")
model = AutoModelForCausalLM.from_pretrained("gpt2")

prompt = "Artificial Intelligence will revolutionize education because"

inputs = tokenizer.encode(prompt, return_tensors="pt")

outputs = model.generate(
    inputs,
    max_length=50,
    do_sample=True,
    temperature=1.2,
    top_p=0.9,
    num_return_sequences=3
)

for i, output in enumerate(outputs):
    print(f"Output {i + 1}: {tokenizer.decode(output, skip_special_tokens=True)}")
```

- Parameters and their values: max_length=50, temperature=1.2, top_p=0.9, do_sample=True, num_return_sequences=3
- Output received:

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention_mask` to obtain reliable results.

Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Output 1: Artificial Intelligence will revolutionize education because that is just how things are designed to be. In other words, if people are going to make progress with our technology it means something. If you think that you are going to learn everything new about you from

Output 2: Artificial Intelligence will revolutionize education because we are starting to see big changes going on.

Today, there are no other means for teachers to understand what a student is thinking, what their own values are, where they came from. And even

Output 3: Artificial Intelligence will revolutionize education because we will be able to create the next generation of researchers and invent new things that we're passionate about to try and make their work better. This would create a culture, or more specifically a new way of teaching
```

- Explanation/Learning: The warning about missing `attention_mask` and `pad_token_id` suggests that results may be unreliable for padded inputs. The higher temperature (1.2) increases randomness and creativity in output, which is

evident in the more diverse and exploratory responses. Setting `pad_token_id` to the EOS token helps mitigate issues during generation.

Configuration-7:

```
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("gpt2")
model = AutoModelForCausalLM.from_pretrained("gpt2")

prompt = "The future of artificial intelligence is"

inputs = tokenizer.encode(prompt, return_tensors="pt")

outputs = model.generate(
    inputs,
    max_length=70,
    do_sample=True,
    temperature=0.7,
    top_p=0.9,
    num_return_sequences=2,
    pad_token_id=tokenizer.eos_token_id
)

for i, output in enumerate(outputs):
    print(f"Output {i + 1}: {tokenizer.decode(output, skip_special_tokens=True)}")
```

- Parameters and their values:max_length=70, temperature=0.7, top_p=0.9, do_sample=True, num_return_sequences=2, pad_token_id=tokenizer.eos_token_id
- Output received:

```
Output 1: The future of artificial intelligence is being decided by a combination of high-stakes competition, government regulation and the sheer speed at which people are embracing the technology. It's a good time to be a bit optimistic.

The problem with Artificial Intelligence

If you're a software engineer, you probably know that when you're writing code, you're

Output 2: The future of artificial intelligence is bright. We're about to find out.

Image copyright Getty Images Image caption The research, led by Professor David Nacke of the University of Manchester, has been published in the journal Science

The results are promising: artificial intelligence will be able to recognise people's faces, for example, and could be
```

- Explanation/Learning: The lower temperature (0.7) results in more focused and coherent text, while still allowing creativity due to sampling (`do_sample=True`). The output maintains structure and logical flow, ideal for informative or analytical content. Specifying `pad_token_id` ensures reliable padding handling, avoiding generation issues.

Configuration-8:

```python
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("gpt2")
model = AutoModelForCausalLM.from_pretrained("gpt2")

prompt = "The future of artificial intelligence is"

inputs = tokenizer.encode(prompt, return_tensors="pt")

outputs = model.generate(
    inputs,
    max_length=70,
    num_beams=2,
    num_return_sequences=2,
    early_stopping=True,
    pad_token_id=tokenizer.eos_token_id,
    top_p=0.9,
    top_k=50,
    repetition_penalty=1.2,
    no_repeat_ngram_size=3
)

for i, output in enumerate(outputs):
    print(f"Output {i + 1}: {tokenizer.decode(output, skip_special_tokens=True)}")
```

- Parameters and their values: max_length=70, temperature=1.0, top_p=0.9, do_sample=False, num_return_sequences=2, pad_token_id=tokenizer.eos_token_id, repetition_penalty=1.2, no_repeat_ngram_size=2
- Output received:

```
/usr/local/lib/python3.11/dist-packages/transformers/generation/configuration_utils.py:633: UserWarning: `do_sample` is set to `False`. However, `top_p` is set to `0.9` -- this flag is only used in sample-based generation modes. You should set `do_sample=True` or unset `top_p`.

  warnings.warn(

Output 1: The future of artificial intelligence is uncertain, but it's clear that we're going to have a lot more of it in the next few years than we've ever had before.

"I think we're on the cusp of a big breakthrough in artificial intelligence," he said. "It's going to be very interesting to see how it evolves
```

```
Output 2: The future of artificial intelligence is uncertain, but it's clear that we're going to have a lot more of it in the next few years than we've ever had before.

"I think we're on the cusp of a big breakthrough in artificial intelligence," he said. "It's going to be very interesting to see what happens with
```

- Explanation/Learning: Here, `do_sample=False` triggers **greedy decoding**, but `top_p=0.9` is still present, causing a warning since it's only meaningful for sampling. The `repetition_penalty=1.2` and `no_repeat_ngram_size=2` help reduce repetitive phrases. The generation is more deterministic but still avoids redundant content, producing coherent and moderately creative outputs.

Post Lab Questions:

1. **What is a pre-trained language model? Give two examples.**

ANS.: A **pre-trained language model** is a type of artificial intelligence system that has already been trained on a large corpus of text data before being used for specific tasks. These models are trained using massive datasets—ranging from books, articles, and websites—to learn the statistical structure of human language, including grammar, syntax, context, semantics, and even some reasoning abilities. The key benefit of pre-trained models is that they can be fine-tuned for downstream tasks (like translation, question answering, or sentiment analysis) with much less data and time compared to training a model from scratch.

These models rely heavily on **deep learning architectures**, particularly **transformers**, which are designed to handle sequences of data (like text) and learn contextual relationships between words. During pre-training, the model usually performs unsupervised tasks such as predicting the next word in a sentence or filling in missing words, thereby learning linguistic patterns without needing labeled data.

Once pre-trained, these models can be adapted for specific natural language processing (NLP) tasks through a process called **fine-tuning**, which involves training the model on a smaller, task-specific dataset. This fine-tuning adjusts the model's parameters slightly so that it performs better for a particular application, such as summarizing articles or detecting spam.

Two popular examples of pre-trained language models:

- **GPT-2 (Generative Pre-trained Transformer 2)**: Developed by OpenAI, GPT-2 is a unidirectional transformer-based model. It is capable of generating human-like text and completing prompts in a way that appears intelligent and context-aware. GPT-2 was trained on 40GB of internet text and can generate

coherent paragraphs, answer questions, and even write stories or code based on an input prompt.

- **BERT (Bidirectional Encoder Representations from Transformers)**: Created by Google, BERT takes a different approach by using a bidirectional transformer architecture. This means it looks at both the left and right context of a word to understand its meaning. BERT is particularly useful for tasks that require a deep understanding of sentence context, such as question answering and named entity recognition.

These models form the backbone of modern NLP and are widely used in academia and industry due to their effectiveness and versatility.

2. **How does GPT-2 generate text?**

ANS. : GPT-2 (Generative Pre-trained Transformer 2) is a transformer-based language model that generates human-like text through a process known as **autoregressive text generation**. This means the model generates one word (or token) at a time, with each word depending on the ones that came before it. It uses a deep neural network architecture composed of multiple transformer layers to achieve this.

Here's a detailed breakdown of the process:

1. Input Encoding (Tokenization):

The model starts with a user-defined prompt. This prompt is first converted into a series of tokens using a tokenizer. Tokens are numerical representations of words or subwords that the model can understand. For instance, the sentence "AI will change education" might be tokenized into something like `[15496, 428, 914, 5342]`.

2. Embedding:

Each token is passed through an embedding layer that transforms it into a dense vector. These vectors capture information about the token's meaning and its position in the sequence.

3. Transformer Layers:

GPT-2 consists of multiple transformer decoder layers, each of which uses **self-attention mechanisms**. These mechanisms allow the model to weigh the importance of different words in the input sequence. For instance, in the sentence "The cat sat on the mat," the word "sat" would pay more attention to "cat" than "mat" for understanding the action.

Since GPT-2 is **unidirectional**, it only looks at the past context (i.e., previous words) and not future words while generating the next one.

4. Text Generation (Decoding):

After processing the input, the model generates a probability distribution over the vocabulary for the next token. It then selects a token based on a decoding strategy:

- **Greedy search** selects the token with the highest probability.
- **Top-k sampling** selects the next token randomly from the top-k most likely tokens.
- **Top-p (nucleus) sampling** chooses from the smallest set of tokens whose cumulative probability exceeds a threshold `p`.
- **Beam search** keeps multiple hypotheses and chooses the one with the highest overall probability.

5. Iterative Generation:

The selected token is added to the input, and the process repeats until a stopping criterion is met—like reaching a maximum length or generating a special end-of-sequence token.

Overall, GPT-2 can generate highly coherent, contextually relevant, and sometimes even creative text, especially when guided by well-crafted prompts and decoding settings.

3. What is the role of prompt engineering in Generative AI?

ANS. : Prompt engineering is a technique used to guide and control the behavior of generative models, especially large language models like GPT-2, GPT-3, and GPT-4. Since these models don't "understand" language in the way humans do, they respond purely based on patterns they've seen during training. Therefore, the way a prompt is phrased plays a crucial role in determining the quality, accuracy, and relevance of the model's output.

In simple terms, prompt engineering is the art and science of crafting effective input prompts to "steer" the model toward producing desired responses.

Why is it important?

- Generative models are sensitive to context, and even minor changes in the prompt can lead to drastically different outputs.
- Without a well-designed prompt, the model may produce outputs that are vague, irrelevant, or incoherent.
- Good prompts can help the model perform tasks like summarization, translation, programming, or even solving logical problems—without retraining the model.

Types of Prompt Engineering Techniques:

1. Instruction-based prompting – Giving clear instructions like “Summarize the following article” helps the model know what is expected.

2. Few-shot prompting – Providing a few examples along with the input so the model learns from context.

3. Chain-of-thought prompting – Encouraging the model to explain its reasoning step-by-step.

4. Zero-shot prompting – Asking the model to perform a task without examples, relying only on well-worded instructions.

In the context of GPT-2:

Since GPT-2 was not explicitly trained for tasks like Q&A or summarization, prompt engineering becomes especially important to "trick" or "guide" the model to behave in a particular way. For example:

- Poor prompt: "Education change."
- Better prompt: "Artificial Intelligence will revolutionize education because..."

The second one sets up a narrative structure, allowing GPT-2 to continue with a meaningful completion.

In summary, prompt engineering is crucial for unlocking the full potential of generative AI models, enabling them to behave like flexible, intelligent assistants for a wide range of tasks.

4. List three applications of Generative AI in NLP.

ANS.: Generative AI, particularly transformer-based language models, has revolutionized Natural Language Processing (NLP) by enabling machines to generate and manipulate human-like text. Below are three key applications where generative AI plays a transformative role:

---

1. Text Generation and Content Creation

Generative AI can write entire articles, social media posts, poems, scripts, and even code. It is used in:

- Creative writing: Assisting writers in brainstorming or generating plot ideas.
- Marketing copy generation: Writing product descriptions or ad content.
- News automation: Generating routine news reports like sports or weather updates.

Tools like ChatGPT, Jasper, and Writesonic use models like GPT-3 and GPT-4 to assist in content creation, often indistinguishable from human-written text.

---

2. Conversational AI and Chatbots

Generative models are at the heart of intelligent chatbots and virtual assistants. Unlike rule-based bots that rely on fixed responses, generative chatbots:

- Engage in dynamic conversations.
- Understand context and tone.
- Answer follow-up questions or clarify user intent.

They are widely used in customer service, mental health support (e.g., Woebot), and educational tutoring platforms.

---

3. Text Summarization and Paraphrasing

Generative AI can understand a long piece of text and produce a shorter version while retaining the core meaning. It is applied in:

- News summarization: Providing brief versions of long articles.
- Academic and legal research: Summarizing lengthy documents or papers.
- Paraphrasing tools: Helping users rewrite content in different tones or complexity levels.

Modern summarization tools often use models like T5 or BART, which are trained specifically for sequence-to-sequence generation tasks.

Conclusion: Thus, from this experiment, we clearly understood how decoding strategies and hyper-parameters influence GPT-2's text creativity, coherence, and output quality.