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

Batch: B2 **Roll No.: 16010122221**

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?)

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

Parameters and their values: High temperature(value:1.5)

Output received: 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 teachers and students who have been trained in the field.

Explanation/Learning: Introduces more randomness, allowing for more diverse and unexpected word choices. Output may become more creative and varied, but can also lose coherence at times.

Parameters and their values: Low temperature(value:0.5)

Output received: 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 teachers and students who have been trained in the field.

The new generation of students will be able to

Explanation/Learning: The model's predictions become more focused on the highest-probability tokens. Output tends to be more conservative, coherent, and sometimes repetitive. Suitable when you need precise and predictable content.

Parameters and their values: Smaller top_k(value:30)

Output received: Artificial Intelligence will revolutionize education because it is now able to learn, not only from humans, but also from the human race. This is in addition to the advances in computer algorithms and algorithms for social and environmental control, and it will also give

Explanation/Learning: The sampling is limited to the very top choices, leading to more predictable and focused outputs. Reduces randomness and tends toward repetitive or safe responses.

Parameters and their values: Higher top_p(value:0.9)

Output received: Artificial Intelligence will revolutionize education because it's much easier to make intelligent decisions. We need smart machines that can do the job in a single day, for example. Smart machines will be as good at answering questions as humans, and they will also

Explanation/Learning: Allows more diverse tokens to be considered, which can result in more creative outputs. There's a higher chance of including less probable (and sometimes unexpected) words, which can reduce overall consistency.

Parameters and their values: Higher num_beams(value:5)

Output received: Artificial Intelligence will revolutionize education because it will make it possible for students to learn about the world around them. It will also make it possible for students to learn about the world around them. It will also make it possible for students to learn about

Explanation/Learning: Often yields more refined and coherent outputs because the model evaluates multiple alternatives. It can be slower and less diverse since it focuses on improving output quality.

Parameters and their values: Shorter max_length(value:30)

Output received: 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 teachers and

Explanation/Learning: Produces concise outputs. Can help focus the response but might not be sufficient for detailed content.

Parameters and their values: Longer max_length(value:100)

Output received: 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 teachers and students who have been trained in the field.

The new generation of students will be able to learn from the best teachers and students who have been trained in the field.

The new generation of students will be able to learn from the best teachers and students who have been trained in the field.

The new generation of students will be

Explanation/Learning: Permits the generation of extended, more elaborated text.

May result in outputs that wander off-topic or become repetitive if not controlled with other parameters.

Parameters and their values: do_sample (value: True)

Output received: Artificial Intelligence will revolutionize education because of a revolutionary trend in AI research:

The new technology will allow us to create amazing things we could never have imagined existed before

But what happens when artificial intelligence learns something and doesn't understand?

Explanation/Learning: Introduces randomness, meaning running the generation multiple times can produce different outputs for the same prompt. Typically results in more creative and varied text.

Parameters and their values: do_sample (value: False)

Output received: 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 teachers and students who have been trained in the field. The new generation of students will be able to

Explanation/Learning: Produces consistent, repeatable outputs by always selecting the highest probability token. Limits creativity but enhances reproducibility and coherence.

Post Lab Questions:

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

   Ans: A pre-trained language model is a neural network model that has been trained on a large corpus of text and can be fine-tuned for specific language tasks. Examples include GPT-2 and BERT.

2. How does GPT-2 generate text?

   Ans: GPT-2 generates text by taking an initial prompt, tokenizing it, and then predicting the next token repeatedly until reaching a maximum length or a defined stopping condition. This prediction is based on the statistical likelihood learned during its pre-training.

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

Ans: Prompt engineering involves designing and tuning the input prompt to steer the model's responses. Effective prompts help in achieving more relevant and coherent outputs tailored to the desired application.

4. List three applications of Generative AI in NLP.

Ans: Common applications include chatbots, automated story writing, and summarization tools.

Conclusion: This experiment demonstrated that fine-tuning GPT-2's hyper-parameters such as temperature, top-k, top-p, max_length, do_sample, and others can significantly influence the balance between creativity and coherence in generated text. By testing different configurations, we learned how each parameter contributes to the model's output, enabling controlled experimentation and optimization for diverse natural language generation tasks.