**Batch: A-3 Roll No.: 16010122104**

**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 Colaband create a new notebook**.**
2. Copy and run the following base codeto 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))*

*-------------------------------------------------------------------------------------------------------*

1. 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
2. Run at least 5 different configurations, each with a new combination of hyper-parameters.
3. 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:** max\_length=50 (default greedy decoding)

**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:**

Greedy decoding always picks the most likely next word. While safe and coherent, it tends to be repetitive and lacks creativity.

**Parameters and their values:** temperature=1.5, do\_sample=True, max\_length=50

**Output received:**

Artificial Intelligence will revolutionize education because it creates opportunities, creating innovative knowledge-sharing and self-discipline programs in high...

**Explanation/Learning:**

Raising temperature increases creativity and randomness. The text is more diverse but slightly less logical and focused.

**Parameters and their values:** do\_sample=True, top\_k=50, max\_length=50

**Output received:**

Artificial Intelligence will revolutionize education because it will allow schools to take on increasingly complex challenges, including how to develop new and...

**Explanation/Learning:**

Top-k sampling limits token choice to the top 50 most probable. The result is structured and sensible while avoiding repetition.

**Parameters and their values:** do\_sample=True, top\_p=0.9, max\_length=50

**Output received:**

Artificial Intelligence will revolutionize education because it can transform traditional educational institutions and give students a place to learn. By leveraging...

**Explanation/Learning:**

Top-p (nucleus sampling) gives balanced and naturally flowing results. It improves quality while still allowing creativity.

**Parameters and their values:** num\_beams=5, early\_stopping=True, max\_length=50

**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...

**Explanation/Learning:**

Beam search improves fluency and coherence by evaluating multiple word sequences. It’s good for quality but may get repetitive.

**Parameters and their values:** do\_sample=True, top\_k=50, top\_p=0.9, temperature=1.0, repetition\_penalty=1.2, max\_length=50

**Output received:**

Artificial Intelligence will revolutionize education because, while it isn't possible to train a computer with its basic functions for human usage (like writing)...

**Explanation/Learning:**

Combining multiple sampling strategies and adding a repetition penalty leads to more unique, non-repetitive, and balanced content.

**Post Lab Questions:**

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

A pre-trained language model is an AI model that has been trained on a large corpus of text data before being fine-tuned for specific tasks. It already understands grammar, syntax, and general knowledge from this training. Examples: GPT-2, BERT

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

GPT-2 generates text by predicting the next word (token) based on the context of the previous words using the transformer architecture. It processes input from left to right and generates words one at a time, continuing until a stopping condition is met.

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

Prompt engineering involves carefully designing the input prompt given to a generative model. It plays a key role in guiding the model to produce relevant, high-quality, and task-specific outputs by setting the right context for generation.

**4. List three applications of Generative AI in NLP.**

• Chatbots and Virtual Assistants

• Story and Content Generation

• Text Summarization

**Conclusion:** Introduced to the concept of GPT.