

## **GENERATIVE AI AND ITS APPLICATIONS**

NAME : GREESHMA YP  
SRN : PES2UG22CS207  
SEM : 6  
SEC : A  
DATE : 22-01-2026

### **Cell 1 (Markdown)**

Introduces the environment setup section and the tools that will be used.

### **Cell 2 (Markdown)**

Explains Hugging Face

A platform that hosts pretrained AI models

Often called the “GitHub of AI”

Used to share and reuse models

### **Cell 3 (Markdown)**

Explains the transformers library

Connects Hugging Face models to Python

Allows downloading and running models easily

### **Cell 4 (Markdown)**

Explains the pipeline() function:

High-level API

Handles preprocessing, model inference, and postprocessing automatically

### **Cell 5 (Markdown)**

States that the pipeline function will now be imported.

### **Cell 6 (Code)**

```
from transformers import pipeline, set_seed, GPT2Tokenizer
```

Imports:

- pipeline → run AI tasks
- set\_seed → make results reproducible
- GPT2Tokenizer → convert text into tokens

```
[6] ✓ 0s set_seed(42)
```

Step 3: Fast Model ( distilgpt2 )

Let's see how the smaller model performs.

```
# Initialize the pipeline with the specific model
fast_generator = pipeline('text-generation', model='distilgpt2')

# Generating text
output_fast = fast_generator(prompt, max_length=50, num_return_sequences=1)
print(output_fast[0]['generated_text'])
```

```
! /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 Colab and restart your session.
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.
WARNING:tensorflow:
config.json: 100%|#####| 782762 [0:00<0:00, 74.36kb/s]
model.safetensors: 100%|#####| 803870208 [0:00<0:00, 15468kb/s]
generation_config.json: 100%|#####| 1247124 [0:00<0:00, 6.16kb/s]
tokenizer_config.json: 100%|#####| 263263 [0:00<0:00, 2.87kb/s]
vocab.json: 100%|#####| 136471484 [0:00<0:00, 1338kb/s]
merges.txt: 100%|#####| 458264 [0:00<0:00, 1.47kb/s]
tokenizer.json: 100%|#####| 138887384 [0:00<0:00, 1.43kb/s]
Device set to use cuda:0
Truncation was not explicitly activated but 'max_length' is provided a specific value, please use 'truncation=True' to explicitly truncate examples to max length. Defaulting to 'longest_first' truncation strategy. If you encode pairs of sequences (BPE-style) with the tokenizer you can select this strategy more precisely by providing a specific strategy to 'truncation'.
Setting 'pad_token_id' to 'eos_token_id':150256 for open-end generation.
Both 'max_new_tokens' (450) and 'max_length' (49) seem to have been set. 'max_new_tokens' will take precedence. Please refer to the documentation for more information. (https://huggingface.co/docs/transformers/main/en/main_classes/text_generation)
Generation AI is a revolutionary technology that can take on the task of finding, learning, and learning in a given environment.
```

Step 4: Standard Model ( gpt2 )

Now let's try the standard model.

```
smart_generator = pipeline('text-generation', model='gpt2')

output_smart = smart_generator(prompt, max_length=50, num_return_sequences=1)
print(output_smart[0]['generated_text'])
```

```
config.json: 100%|#####| 803882 [0:00<0:00, 74.36kb/s]
model.safetensors: 100%|#####| 803870208 [0:00<0:00, 15468kb/s]
generation_config.json: 100%|#####| 1247124 [0:00<0:00, 6.16kb/s]
tokenizer_config.json: 100%|#####| 263263 [0:00<0:00, 2.86kb/s]
vocab.json: 100%|#####| 136471484 [0:00<0:00, 1.47kb/s]
merges.txt: 100%|#####| 458264 [0:00<0:00, 1.43kb/s]
tokenizer.json: 100%|#####| 138887384 [0:00<0:00, 1.43kb/s]
Device set to use cuda:0
Truncation was not explicitly activated but 'max_length' is provided a specific value, please use 'truncation=True' to explicitly truncate examples to max length. Defaulting to 'longest_first' truncation strategy. If you encode pairs of sequences (BPE-style) with the tokenizer you can select this strategy more precisely by providing a specific strategy to 'truncation'.
Setting 'pad_token_id' to 'eos_token_id':150256 for open-end generation.
Both 'max_new_tokens' (450) and 'max_length' (49) seem to have been set. 'max_new_tokens' will take precedence. Please refer to the documentation for more information. (https://huggingface.co/docs/transformers/main/en/main_classes/text_generation)
Generation AI is a revolutionary technology that enables a wide range of intelligent systems to work independently from one another. It introduces a new way of thinking about AI and provides a new paradigm for the development of intelligent AI.
In this article, we will discuss the main features of the new AI platform, and how it can be used to help us create a world that will improve our lives for the better.
1. How Can I Use It?
The concept of AI is not new. It has been used by many people to measure their mental health and health-related behaviors, and as a tool for medical research. It has been used by many of us to track and report on our mental health.
It is based on the premise that AI is a way for humans to move towards a more efficient way of thinking, and therefore, a better way of living.
In this article, we will explain what AI can do.
What does it do
In this article, we will explain how all of our cognitive and emotional systems interact with the AI platform. The main features of AI are:
A new way of thinking about AI
A new paradigm for the development of intelligent AI
A new way of thinking about mental health and health-related behaviors
```

```
[6] ✓ 0s set_seed(55)
```

```

Step 4: Standard Model (gpt2)
Now let's try the standard model.

In [ ]:
user_generator = pipeline('text-generation', model='gpt2')

# Generate text
output_text = user_generator(prompt, max_length=50, num_return_sequences=1)
print(output_text[0].generator_text[1])

config.json: 100% 454448 [0:00:00.00, 67.84MB]
model.safetensors: 100% 548033984 [0:00:00.00, 10.94MB]
generation_config.json: 100% 124 [0:00:00.00, 0.00MB]
tokenizer_config.json: 100% 28.00KB [0:00:00.00, 2.77MB]
vocab.json: 100% 1.56MB [0:00:00.00, 5.43MB]
merges.txt: 100% 456408 [0:00:00.00, 2.75MB]
tokenizer.json: 100% 1.38MB [0:00:00.00, 6.24MB]

Device set to use cuda:0
Truncation was not explicitly activated but 'max_length' is provided a specific value, please use 'truncation=True' to explicitly truncate examples to max length. Defaulting to 'longest_first' truncation strategy. If you encode pairs of sequences (BPE-style) with the tokenizer you can select this strategy more precisely by provid
Setting 'max_length' to 50. You can set it to 0 for open-ended generation.
Both 'max_new_tokens' (40) and 'max_length' (50) seem to have been set. 'max_new_tokens' will take precedence. Please refer to the documentation for more information. (https://huggingface.co/docs/transformers/main_classes/text_generation)
Generating text is a revolutionary technology that has the potential to transform the way we collect and share information. It's a new way of helping children and their parents improve their lives.
The New York Times, July 29, 2021
The new technology, which is being proposed by the U.S. government, will help parents search for information related to their children's education. The technology is being developed with the help of the National Center for Education Statistics, which collects information about school performance, such as the number of students who
The Economist, July 29, 2021
The U.S. government is making progress in developing a universal program designed to help poor children learn to cope with school-related problems. The program will provide money to parents to help them keep their children from getting into trouble. The program will help parents search for information related to their children's ac

```

```

Step 3: Fast Model (distilgpt2)
Let's see how the smaller model performs.

In [ ]:
# Initialize the pipeline with the specific model
fast_generator = pipeline('text-generation', model='distilgpt2')

# Generate text
output_text = fast_generator(prompt, max_length=50, num_return_sequences=1)
print(output_text[0].generator_text[1])

config.json: 100% 762762 [0:00:00.00, 28.36MB]
model.safetensors: 100% 103603008 [0:00:00.00, 1.94MB]
generation_config.json: 100% 124 [0:00:00.00, 0.00MB]
tokenizer_config.json: 100% 28.00KB [0:00:00.00, 2.76MB]
vocab.json: 100% 1.56MB [0:00:00.00, 5.43MB]
merges.txt: 100% 456408 [0:00:00.00, 2.75MB]
tokenizer.json: 100% 1.38MB [0:00:00.00, 6.24MB]

Device set to use cuda:0
Truncation was not explicitly activated but 'max_length' is provided a specific value, please use 'truncation=True' to explicitly truncate examples to max length. Defaulting to 'longest_first' truncation strategy. If you encode pairs of sequences (BPE-style) with the tokenizer you can select this strategy more precisely by pro
Setting 'max_length' to 50. You can set it to 0 for open-ended generation.
Both 'max_new_tokens' (40) and 'max_length' (50) seem to have been set. 'max_new_tokens' will take precedence. Please refer to the documentation for more information. (https://huggingface.co/docs/transformers/main_classes/text_generation)
Generating text is a revolutionary technology that has the potential to revolutionize the way people interact with the world, and that is why we are launching our first AI class on October 1st at the AI World Congress, in the US.

```

## Cell 7 (Markdown)

Mentions importing additional utilities for NLP and file handling.

## Cell 8 (Code)

import os

import nltk

os → file system operations

nltk → traditional NLP tasks like tokenization and POS tagging

## Cell 9 (Markdown)

Introduces the text file (unit 1.txt) that will act as course material / knowledge base.

## Cell 10 (Code)

file\_path = "unit 1.txt"

Stores the file path of the course content.

## Cell 11 (Markdown)

Explains that the file will be read and stored as text.

### **Cell 12 (Code)**

try:

```
with open(file_path, "r", encoding="utf-8") as f:
```

```
    text = f.read()
```

```
    print("File loaded successfully!")
```

except FileNotFoundError:

```
    print(f"Error: '{file_path}' not found.")
```

Reads the text file safely

Handles error if file is missing

### **Cell 13 (Markdown)**

Mentions previewing the first part of the data.

### **Cell 14 (Code)**

```
print(text[:500])
```

Displays the first 500 characters to verify correct data loading.

### **Cell 15 (Markdown)**

Introduces Generative AI model comparison:

Small vs large models

### **Cell 16 (Markdown)**

Explains random seed:

Ensures reproducible outputs

### **Cell 17 (Code)**

```
set_seed(42)
```

Fixes randomness so results remain the same every run

### **Cell 18 (Markdown)**

Introduces the prompt for text generation.

### **Cell 19 (Code)**

`prompt = "Generative AI is a revolutionary technology that"`

Defines input sentence for the model to complete.

#### **Cell 20 (Markdown)**

Introduces DistilGPT-2, a smaller and faster model.

#### **Cell 21 (Code)**

```
fast_generator = pipeline('text-generation', model='distilgpt2')
```

Creates a text generation pipeline using a lightweight model.

#### **Cell 22 (Markdown)**

Introduces the standard GPT-2 model.

#### **Cell 23 (Code)**

```
smart_generator = pipeline('text-generation', model='gpt2')
```

Uses a larger model for better quality text generation.

#### **Cell 24 (Markdown)**

Analysis prompt:

Compare coherence, relevance, and fluency of both models

#### **Cell 25 (Markdown)**

Starts NLP fundamentals section.

#### **Cell 26 (Markdown)**

Explains tokenization:

Converts text → tokens → numbers

#### **Cell 27 (Code)**

```
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
```

Loads GPT-2 tokenizer.

#### **Cell 28 (Markdown)**

Introduces sample sentence.

#### **Cell 29 (Code)**

```
sample_sentence = "Transformers revolutionized NLP."
```

### **Cell 30 (Markdown)**

Mentions tokenization step.

### **Cell 31 (Code)**

```
tokens = tokenizer.tokenize(sample_sentence)
```

Splits sentence into tokens.

### **Cell 32 (Markdown)**

Mentions converting tokens to IDs.

### **Cell 33 (Code)**

```
token_ids = tokenizer.convert_tokens_to_ids(tokens)
```

Converts tokens into numeric IDs used by models.

### **Cell 34 (Markdown)**

Explains POS tagging:

Identifies nouns, verbs, adjectives, etc.

### **Cell 35 (Code)**

```
nltk.download('averaged_perceptron_tagger')
```

```
nltk.download('punkt')
```

Downloads required NLP resources.

### **Cell 36 (Markdown)**

Mentions tagging sentence.

### **Cell 37–38 (Code)**

```
tokens = nltk.word_tokenize(sample_sentence)
```

```
pos_tags = nltk.pos_tag(tokens)
```

Performs Part-of-Speech tagging.

### **Cell 39 (Markdown)**

Introduces Named Entity Recognition (NER).

**Cell 40 (Code)**

```
pipeline("ner", model="dbmdz/bert-large-cased-finetuned-conll03-english")
```

Loads a BERT model trained for NER.

**Cell 41–42 (Code)**

Runs NER on text

Extracts entities like Person, Organization, Location

Filters high-confidence entities

**Cell 43 (Markdown)**

Starts advanced applications section.

**Cell 44 (Markdown)**

Introduces summarization model comparison.

**Cell 45 (Code)**

Provides long transformer-related text for summarization.

**Cell 46–47 (Code)**

Uses DistilBART for fast summarization.

**Cell 48–49 (Code)**

Uses BART-Large for higher quality summarization.

**Cell 50 (Markdown)**

Introduces Question Answering task.

**Cell 51 (Code)**

```
pipeline("question-answering")
```

Loads QA model.

**Cell 52–53 (Code)**

Asks questions and extracts answers from text.

**Cell 54 (Markdown)**

Introduces Masked Language Modeling.

**Cell 55 (Code)**

```
pipeline("fill-mask")
```

Loads BERT fill-in-the-blank model.

**Cell 56–57 (Code)**

Predicts missing word in a sentence and prints probabilities.