Day 3 — Build a Text Generator (Mini ChatGPT)

🧩 What You Are Building Today

- A Mini ChatGPT / Text Generator that:

1. Takes user input (prompt)
2. Sends it to a Transformer model
3. Generates AI text as output

```
!pip install transformers --quiet

from transformers import pipeline
generator = pipeline("text-generation", model="distilgpt2")
```

```
WARNING:torchao.kernel.intmm:Warning: Detected no triton, on systems without Tri
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserW
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 (
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access publ
  warnings.warn(
config.json: 100%  762/762 [00:00<00:00, 50.7kB/s]
model.safetensors: 100%  353M/353M [00:02<00:00, 253MB/s]
generation_config.json: 100%  124/124 [00:00<00:00, 8.83kB/s]
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 2.51kB/s]
vocab.json: 100%  1.04M/1.04M [00:00<00:00, 4.18MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 2.75MB/s]
tokenizer.json: 100%  1.36M/1.36M [00:00<00:00, 4.09MB/s]
Device set to use cpu
```

```
def generate_text(prompt):
    result = generator(
        prompt,
        max_new_tokens=80,
        temperature=0.9,
        top_p=0.95,
        num_return_sequences=1
```

```
    )
    return result[0]["generated_text"]

prompt = "Artificial Intelligence will change the world by"
output = generate_text(prompt)

print("AI Output:\n")
print(output)
```

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

Artificial Intelligence will change the world by moving to a new approach to it.




The company is now working on a "cloud platform," which will allow customers to

The cloud platform is already using Google Drive as a way
```

```
user_prompt = input("Enter your prompt: ")
response = generate_text(user_prompt)

print("\n🤖 AI Response:\n")
print(response)
```

```
Enter your prompt: life
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

🤖 AI Response:

life A very popular, non-commercial, non-commercial, non-commercial, non-commerc
```

```
def generate_multiple(prompt):
    results = generator(
        prompt,
        max_new_tokens=60,
        temperature=1.0 ,
        num_return_sequences=3
    )
    for i, r in enumerate(results):
        print(f"\n--- Response {i+1} ---")
        print(r["generated_text"])

generate_multiple("i love you ")
```

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

--- Response 1 ---
```

```
i love you !! I know the time has come (and we didn't see some). So I just want
I'm looking forward to the upcoming release but I have no doubt that, from the s

--- Response 2 ---
i love you !!! LOVE you!!! THANKS TO MY KIDS TO ALL OF YOU AND TO THE KIDS. MOST
LIVE TWO BODY!

--- Response 3 ---
i love you !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

1️⃣ How **Text Generators Work Internally** (CORE IDEA)

A text generator **does NOT think**. It **predicts the next word** again and again.

Simple example:

Sentence:

> I love ice

Model asks: 👉 "What word usually comes after `I love ice`?"

From training data, it knows:

- cream (70%)
- skating (10%)
- coffee (5%)
- coding (0.1%)

It **chooses one**, adds it:

> I love ice cream

Then repeats:

> "What comes after `I love ice cream`?"

This repeats **token by token** until stopping.

📌 **That's it.** No brain. No emotions. Just probability.

2️⃣ How **Prompts Flow Into a Model**

Your prompt is just **input text**.

What really happens:

```
Your Prompt (text)
   ↓
Tokenizer (text → numbers)
   ↓
Transformer Model (math + probabilities)
   ↓
Generated Tokens (numbers)
   ↓
Detokenizer (numbers → text)
```

Example:

```
prompt = "AI will replace"
```

Internally becomes:

```
[1012, 4532, 9981]   ← numbers (tokens)
```

Model predicts next token:

```
→ [work]
→ [jobs]
→ [humans]
```

📌 Prompt = **starting context**, not instructions.

3️⃣ What is a **Transformer** (IMPORTANT)

A **Transformer** is the architecture that:

- Understands **context**
- Looks at **all words together**, not one by one
- Uses **Attention**

Attention means:

> "Which words matter most right now?"

Example:

> **The animal didn't cross the road because it was tired**

What is **it**?

- animal ✅
- road ❌

Transformer figures this using **attention weights**.

📌 That's why transformers are powerful.

4️⃣ How Output is Controlled (VERY IMPORTANT)

Now your confusion parts 👇 These are NOT random magic values.

🔥 `temperature` — Creativity Control

Controls **how risky the word choice is**.

Low temperature (0.2 – 0.5)

- Safe
- Repetitive
- Boring
- Most probable word chosen

High temperature (0.9 – 1.2)

- Creative
- Random
- Emotional
- Sometimes nonsense

📌 Example:

```
temperature = 0.3  # strict, logical
temperature = 1.0  # creative
```

🔥 `top_p` — Word Selection Filter

Also called **nucleus sampling**.

Model has MANY word choices. `top_p` says:

> "Only consider words that make up X% probability"

Example:

If `top_p = 0.9` → Ignore weird low-probability words

📌 Helps avoid nonsense while keeping creativity.

🔥 `max_new_tokens` — Length Control

This is **how many words/tokens AI can add**.

```
max_new_tokens = 50
```

Means:

> AI can generate **up to 50 tokens AFTER your prompt**

If prompt is long, it doesn't matter.

📌 This is better than `max_length` (old method).

5️⃣ Why Your Output Was WEIRD (IMPORTANT)

You saw:

> OpenOffice nonsense text

WHY?

Because:

- `distilgpt2` is **small**
- Trained on **old + noisy data**
- High randomness
- No safety filters

📌 That's expected. Not your fault.

Bigger models = better output.

6️⃣ How a **Basic Chatbot Is Structured**

A chatbot is **NOT special**.

It is just:

```
conversation = ""

while True:
    user_input = input("You: ")
    conversation += "User: " + user_input + "\nAI:"
    response = generator(conversation)
    print(response)
```

📌 Chatbot = **Text generator + memory (conversation history)**

That's ALL.
