```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
!nvidia-smi
```

```
Fri Feb 14 18:14:08 2025
+-----------------------------------------------------------------------------------------
| NVIDIA-SMI 550.54.15              Driver Version: 550.54.15      CUDA Version: 12.4
|-----------------------------------------+------------------------+----------------
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Comput
|                                         |                        |               MI
|=========================================+========================+================
|   0  Tesla T4                       Off |   00000000:00:04.0 Off |
| N/A   36C    P8              8W /   70W |       0MiB /  15360MiB |      0%      Def
|                                         |                        |
+-----------------------------------------+------------------------+----------------

+-----------------------------------------------------------------------------------------
| Processes:
|  GPU   GI   CI        PID   Type   Process name                              GPU Me
|        ID   ID                                                               Usage
|=========================================================================================
|  No running processes found
+-----------------------------------------------------------------------------------------
```

```
from transformers import AutoTokenizer

#Load the tokenizer for a specific model (e.g.m GPT-2)
tokenizer = AutoTokenizer.from_pretrained("gpt2")

#Tokenize some input text
text = "Hello, How was your Day?"
tokens = tokenizer(text, return_tensors='pt')
print(tokens)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarnin
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
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 mo
  warnings.warn(
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 783B/s]
config.json: 100%  665/665 [00:00<00:00, 35.0kB/s]
vocab.json: 100%  1.04M/1.04M [00:00<00:00, 1.59MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 1.08MB/s]
tokenizer.json: 100%  1.36M/1.36M [00:00<00:00, 6.17MB/s]
{'input_ids': tensor([[15496,    11,  1374,   373,   534,  3596,    30]]), 'attention
```

```
from transformers import AutoModelForCausalLM

#Load the pre-trained GPT- model
model = AutoModelForCausalLM.from_pretrained("gpt2")

# Generate text
input_ids = tokenizer.encode("indian cricket", return_tensors='pt')
output = model.generate(input_ids, max_length=50)
generated_text = tokenizer.decode(output[0], skip_special_tokens=True)
print(generated_text)
```

```
model.safetensors: 100%  548M/548M [00:02<00:00, 185MB/s]
generation_config.json: 100%  124/124 [00:00<00:00, 11.4kB/s]
The attention mask and the pad token id were not set. As a consequence, you may obser
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
indian cricket team, which has been in the country for over a decade.

The team's captain, Ravi Shankar, has been in the country for over a decade.

The team's captain, Ravi Shankar,
```

```
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load a pre-trained model and tokenizer
model_name = "gpt2"  # You can replace with any other LLM
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

def chunk_text(text, max_length=512):
    """Chunk text into smaller pieces."""
    tokens = tokenizer.encode(text, return_tensors='pt')[0]
    chunks = []
```

```
        for i in range(0, len(tokens), max_length):
            chunk = tokens[i:i + max_length]
            chunks.append(chunk)

        return chunks

    def generate_responses(chunks):
        """Generate responses for each chunk using the LLM."""
        responses = []
        for chunk in chunks:
            input_ids = chunk.unsqueeze(0)  # Add batch dimension
            # Increase max_length to a value greater than or equal to the longest chunk lengt
            output = model.generate(input_ids, max_length=512)  # Generate response
            responses.append(tokenizer.decode(output[0], skip_special_tokens=True))

        return responses

    # Example long text
    long_text = "India " * 5  # Repeat to simulate long text

    # Chunk the text
    chunks = chunk_text(long_text)

    # Generate responses for each chunk
    responses = generate_responses(chunks)

    # Print the responses
    for i, response in enumerate(responses):
        print(f"Response for chunk {i+1}:\n{response}\n")

    #print(responses)
```

```
The attention mask and the pad token id were not set. As a consequence, you may obser
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Response for chunk 1:
India India India India India  (India) India India India India India India India Indi
```

```
    # Introduce to Hugging face
    # Import the model
    # refer for hugging face -->https://huggingface.co/openai-community/gpt2

    from transformers import pipeline, set_seed
    generator = pipeline('text-generation', model='gpt2')
    set_seed(45)
    generator("Hello, I'm an artificail robot model,", max_length=30, num_return_se
```

```
Device set to use cuda:0
Truncation was not explicitly activated but `max_length` is provided a specific value
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': "Hello, I'm an artificail robot model, making small robots with
very limited life and resources. So here we are. I will use my life"},
 {'generated_text': "Hello, I'm an artificail robot model, the first of its kind
from a toy company to have a robotic arm. I'm designing and testing"},
```

```
 {'generated_text': "Hello, I'm an artificail robot model, I've been working on the
model for 3 years now. I'm in the process of drawing what"},
 {'generated_text': "Hello, I'm an artificail robot model, and I have an old, bad
old, great old model for your needs.[1]\n\n"},
 {'generated_text': 'Hello, I\'m an artificail robot model, you see!\n\n"W-W-What?
Why are you wearing it now that you'}]
```

```
from transformers import pipeline, set_seed
generator = pipeline('text-generation', model='gpt2')
set_seed(45)
generator("Hello, explain about indian economy,", max_length=10, num_return_sequences=2)
```

```
Device set to use cuda:0
Truncation was not explicitly activated but `max_length` is provided a specific value
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': 'Hello, explain about indian economy, how you'},
 {'generated_text': 'Hello, explain about indian economy, the reasons'}]
```



```
from transformers import GPT2Tokenizer, GPT2Model
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = GPT2Model.from_pretrained('gpt2')
text = "Replace me by any text you'd like."
encoded_input = tokenizer(text, return_tensors='pt')
output = model(**encoded_input)
print(output)
```

```
                [ 0.5957, -0.2422, -0.1601,  ...,  0.0871,  0.6242,  0.0631]],

               ...,

               [[-0.0194, -0.0276,  0.0886,  ...,  0.0796, -0.0209,  0.0248],
                [-0.7218,  0.9741,  0.7868,  ..., -0.1377, -0.3252, -1.0529],
                [ 0.0102,  0.0130,  0.1943,  ...,  1.0051,  0.9481, -0.4571],
                ...,
                [-1.2697,  1.1965,  1.8222,  ...,  1.2815,  1.1525, -0.2608],
                [-0.9059, -0.1876, -0.2131,  ...,  0.1001,  0.5176, -0.7554],
                [-0.2481,  0.0416, -0.7926,  ...,  0.2645, -0.6107, -0.3649]],

               [[-0.1515, -0.0920,  0.0492,  ..., -0.0616,  0.0336, -0.0914],
                [ 0.1947,  0.3574,  0.4865,  ..., -0.0827, -0.0695,  0.1024],
                [ 0.0617, -0.4696,  0.1419,  ..., -0.5913, -0.3143,  0.7776],
                ...,
                [-0.4784,  1.0185, -0.0705,  ...,  0.2748, -0.4973,  1.3698],
                [-0.8326,  0.6881,  0.1242,  ..., -0.5708,  0.6708,  0.8386],
                [-1.0543, -0.0815,  0.9794,  ...,  0.3561,  0.6065,  0.8012]],

               [[ 0.1127, -0.1414,  0.0995,  ..., -0.1078,  0.0248, -0.1947],
                [ 0.3453, -0.7535,  0.9195,  ...,  0.1146, -0.0401,  0.5830],
                [-0.6160, -0.7786,  1.2499,  ..., -1.0763,  0.0126, -0.6472],
                ...,
                [-1.0815,  0.2212,  0.6810,  ..., -1.4694, -0.5813,  0.5124],
                [-0.5673, -0.7975, -0.1831,  ...,  0.2839,  0.3034,  0.0535],
                [-0.9700,  0.6699, -0.1582,  ...,  0.8679, -0.3234,  1.0039]]]],
              grad_fn=<TransposeBackward0>))), hidden_states=None, attentions=None, cross
```

```
# tensorflow vectorization code

from transformers import GPT2Tokenizer, TFGPT2Model
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
model = TFGPT2Model.from_pretrained('gpt2')
text = "Replace me by any text you'd like."
encoded_input = tokenizer(text, return_tensors='tf')
output = model(encoded_input)
print(output)
```

```
                [ 0.1144134 , -0.85596406,  1.3443873 , ..., -0.96876144,
                 -0.01114578,  0.6124071 ],
                [ 0.37190548, -1.1887642 ,  2.986621  , ...,  1.556443  ,
                  1.056937  ,  1.0003769 ]]]],



              [[[[ 0.06268022, -0.10412533, -0.18609698, ..., -0.29731685,
                   0.26168963, -0.12999761],
                 [ 0.91903317, -0.4094241 ,  0.9731715 , ...,  0.7065386 ,
                  -1.3205703 ,  1.6103195 ],
                 [-0.33137167,  0.70993197,  0.8129847 , ...,  1.2446222 ,
                  -1.0029218 ,  1.6824276 ],
                 ...,
                 [ 0.30730325, -0.17975262,  2.0175369 , ...,  3.858773  ,
                  -1.2388641 ,  0.951818  ],
                 [-1.0282463 ,  0.0809522 ,  1.8490098 , ...,  2.1276493 ,
                  -0.6253651 ,  0.25800744],
                 [-2.1301045 ,  0.1848369 ,  0.63887763, ...,  0.84967697,
                  -2.189411  ,  2.4371755 ]],

                [[ 0.07013769, -0.03663868,  0.04331543, ..., -0.02050136,
                  -0.12257352,  0.18812777],
                 [-0.48283267,  0.03970676,  0.11329718, ...,  0.66455376,
                  -0.41219985, -0.4975902 ],
                 [-0.05595522,  0.5185024 , -0.37963897, ..., -0.03581827,
                  -1.7323874 , -0.59872144],
                 ...,
                 [ 0.36212328, -1.0770136 ,  0.84159803, ..., -1.0863295 ,
                  -1.4621209 ,  1.3165252 ],
                 [-0.41697866, -0.1856038 , -0.22080503, ...,  0.66794723,
                   0.2648093 , -0.73303694],
                 [ 0.93269575,  0.1231287 , -0.25675896, ...,  0.02055706,
                  -0.5632305 , -0.03475542]],

                [[ 0.01024648,  0.0406803 , -0.04272784, ...,  0.0175696 ,
                   0.03244145,  0.05450547],
                 [-0.8473519 , -0.13391528,  0.6198161 , ..., -1.1320072 ,
                  -0.10284051,  0.0237357 ],
```

```
from transformers import pipeline, set_seed
generator = pipeline('text-generation', model='gpt2')

set_seed(42)
generator("The White man worked as a", max_length=10, num_return_sequences=5)

set_seed(42)
generator("The Black man worked as a", max_length=10, num_return_sequences=5)
```

```
Device set to use cuda:0
Truncation was not explicitly activated but `max_length` is provided a specific value
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': 'The Black man worked as a clerk at the warehouse'},
 {'generated_text': 'The Black man worked as a cop in Southwark'},
 {'generated_text': 'The Black man worked as a chef, bartender and'},
 {'generated_text': 'The Black man worked as a housekeeper in the'},
 {'generated_text': 'The Black man worked as a salesman for a local'}]
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.