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

```
Mounted at /content/drive
```

```
!pip install transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.48.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from transformers) (3.17.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.28.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.5.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.24.0->trans
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub<1.0,>=0.2
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2025.1.
```

```
from transformers import AutoTokenizer

# Load the tokenizer for a specific model (e.g., Gpt=2)
tokenizer = AutoTokenizer.from_pretrained("gpt2")

# Tokenize some input text
text = "Hello, how are you?"
tokens = tokenizer(text, return_tensors='pt')

# Print the tokens
print(tokens)
```

```
{'input_ids': tensor([[15496,    11,   703,   389,   345,    30]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1]])}
```

```
from transformers import AutoModelForCausalLM

# Load the pre-trained GPT-2 model
model = AutoModelForCausalLM.from_pretrained("gpt2")

# Generate the text
input_ids = tokenizer.encode("Once upon a time", 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 [00:09<00:00, 126MB/s]

generation_config.json: 100%                                124/124 [00:00<00:00, 6.52kB/s]
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input
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 same as eos token. As a consequence, you may ob
Once upon a time, the world was a place of great beauty and great danger. The world was a place of great danger, and the world was a
```

```
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(chunk):
  """Generate responses for each chunk using the LLM."""
  responses = []
  for chunk in chunks:
    input_ids = chunk.unsqueeze(0) # Add bath dimension
    # Increase max_length to a value greater than or equal to the longest chunk length
    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 = "brief explain about genrative ai " * 50 # 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")
```

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Response for chunk 1:
brief explain about genrative ai brief explain about genrative ai brief explain about genrative ai brief explain about genrative ai

The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Response for chunk 1:
brief explain about genrative ai brief explain about genrative ai brief explain about genrative ai brief explain about genrative ai
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.