



NATURAL LANGUAGE PROCESSING

# TEXT GENERATION



# MODEL ARCHITECTURES

## Autoregressive

Autoregressive models (e.g., GPT-2, T5) predict the next token in a sequence based on the previous ones, generating text one word at a time. This allows for high coherence and control over the generated text.

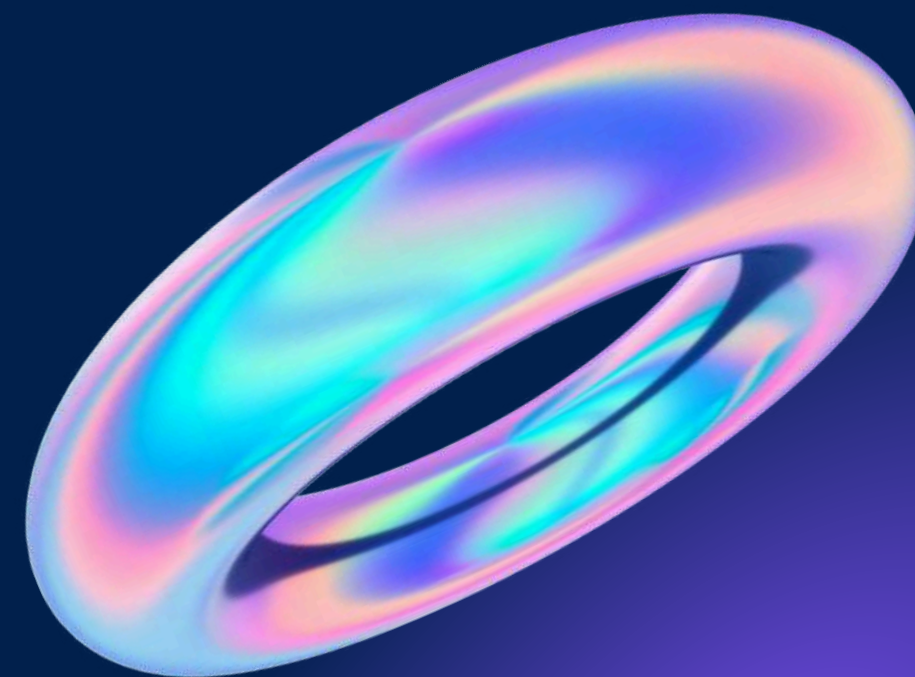
## Non-autoregressive

Non-autoregressive models (e.g., BART) can generate the entire sequence at once, often leading to faster generation and potentially more diverse outputs. However, they may sacrifice some control over the coherence.





**We used GPT-2**





# Why GPT-2 ?

- High Coherence : GPT-2 excels at generating realistic and coherent text, mimicking various writing styles and tones.
- Adaptability: Style and Content Matching: GPT-2 can adapt its output to the style and content of the provided prompt. This allows for diverse and engaging text generation.
- Data Augmentation: Enhancing Datasets: GPT-2 can be used to generate additional text data, which can be helpful in data science and NLP tasks when training data is limited. This can improve the performance of machine learning models

# HIGH COHERENCE MAKES IT IDEAL FOR TASKS LIKE



01

Creative writing:  
Generating poems, scripts,  
musical pieces, or other  
creative text formats.



02

Story continuation:  
Continuing a story based  
on a starting prompt.



03

Dialogue creation: Building  
chatbots or  
conversational AI.





## OTHER ADVANTAGES

Accessibility: GPT-2 is readily available through Hugging Face, making it easy to use and experiment with.

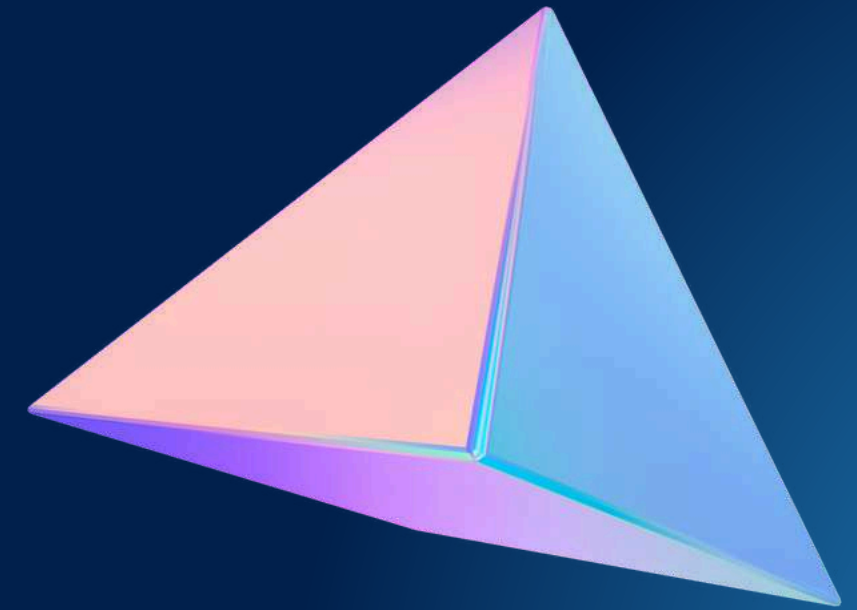
Large Language Model Capabilities: While not the most advanced model anymore, GPT-2 still possesses the general capabilities of large language models, including: Question answering, Summarization and Translation (in specific domains) However, it's important to consider these limitations:

Potential for Bias: GPT-2 was trained on a massive dataset that may contain biases. This can be reflected in its generated text, requiring careful monitoring and potential fine-tuning.

Repetitive or Nonsensical Outputs: When generating long passages, GPT-2 can sometimes produce repetitive or nonsensical text.

# Overall

GPT-2 remains a valuable tool for text generation tasks, particularly when seeking high coherence and adaptability. Its accessibility and general capabilities make it a good starting point for exploring the potential of large language models.





# What are alternatives of GPT-2

- T5
- BART (Bidirectional & Autoregressive Transformer)
- XLNet



# BERT

## Architecture

BERT is based on the Transformer architecture, specifically the encoder part. It considers bidirectional context when encoding input sequences.

## Text Generation

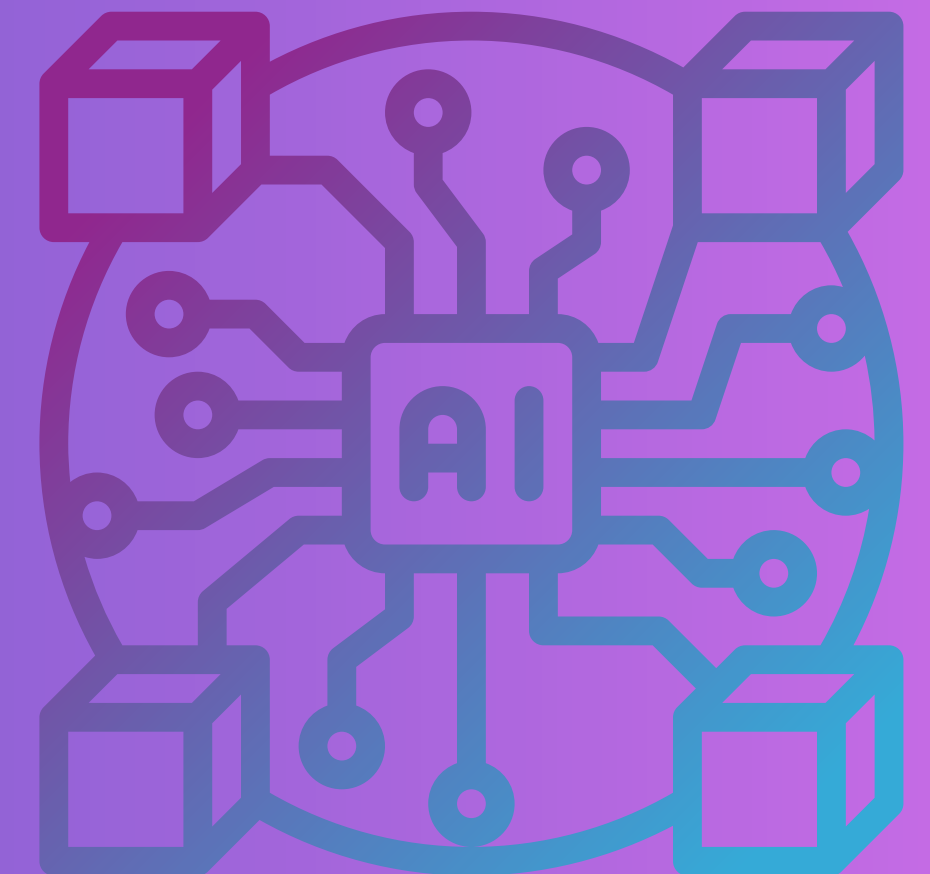
While BERT is primarily designed for tasks like classification and question answering, it can also be used for text generation tasks, although its generation capabilities are not as strong as GPT-2.

## Fine-tuning

BERT can be fine-tuned on downstream tasks, but its primary focus is on tasks like sentence classification and token-level tasks like named entity recognition.

## Conditional Generation

BERT can generate text conditioned on prompts, but its generation capabilities are limited compared to autoregressive models like GPT-2.



# XLNET

## Architecture

XLNet is based on the Transformer architecture like BERT, but it incorporates permutation-based training objectives to capture bidirectional context without relying on masks or segmentation.

## Text Generation

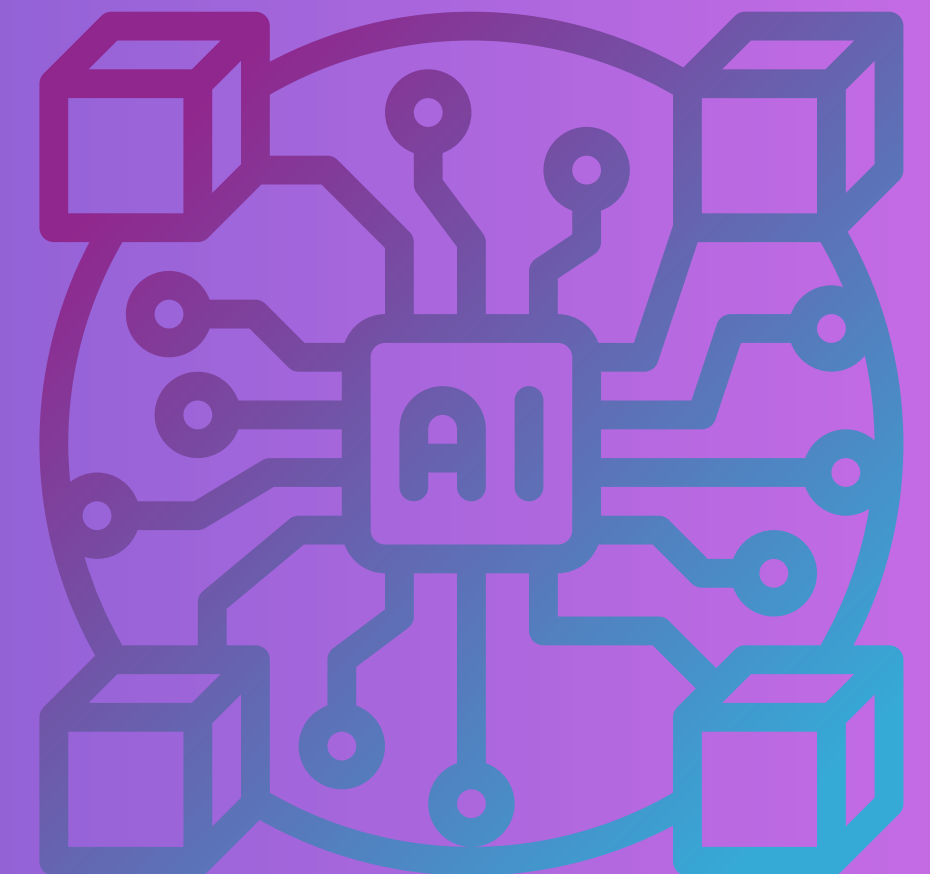
XLNet has strong text generation capabilities, similar to GPT-2, due to its ability to capture bidirectional context effectively.

## Fine-tuning

XLNet can be fine-tuned on specific tasks, similar to BERT and GPT-2, to improve its performance on text generation tasks.

## Conditional Generation

XLNet can generate text conditioned on prompts or input sequences, offering control over the content and style of the generated text.





## Architecture

T5 is based on the Transformer architecture, but it formulates all NLP tasks as text-to-text tasks, where both inputs and outputs are text sequences.

## Text Generation

T5 is designed specifically for text generation tasks and has shown strong performance across various text generation tasks, including summarization, translation, and question answering.

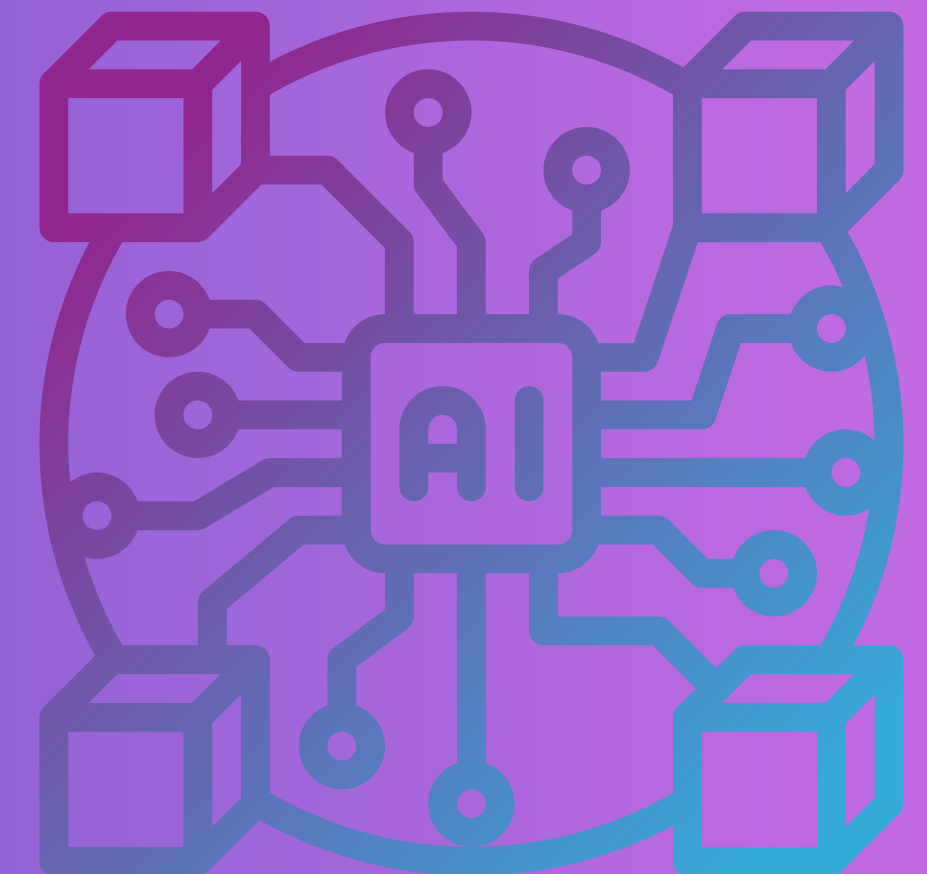
## Fine-tuning

T5 can be fine-tuned on specific tasks, and its text-to-text formulation makes it versatile for various text generation tasks.

## Conditional Generation

T5 can generate text conditioned on prompts or input sequences, offering flexibility in controlling the content and style of the generated text.

# T5



# GPT-2

## Architecture

GPT-2 is based on the Transformer architecture, specifically the decoder part. It generates text autoregressively, predicting the next word in a sequence based on the preceding context.

## Text Generation

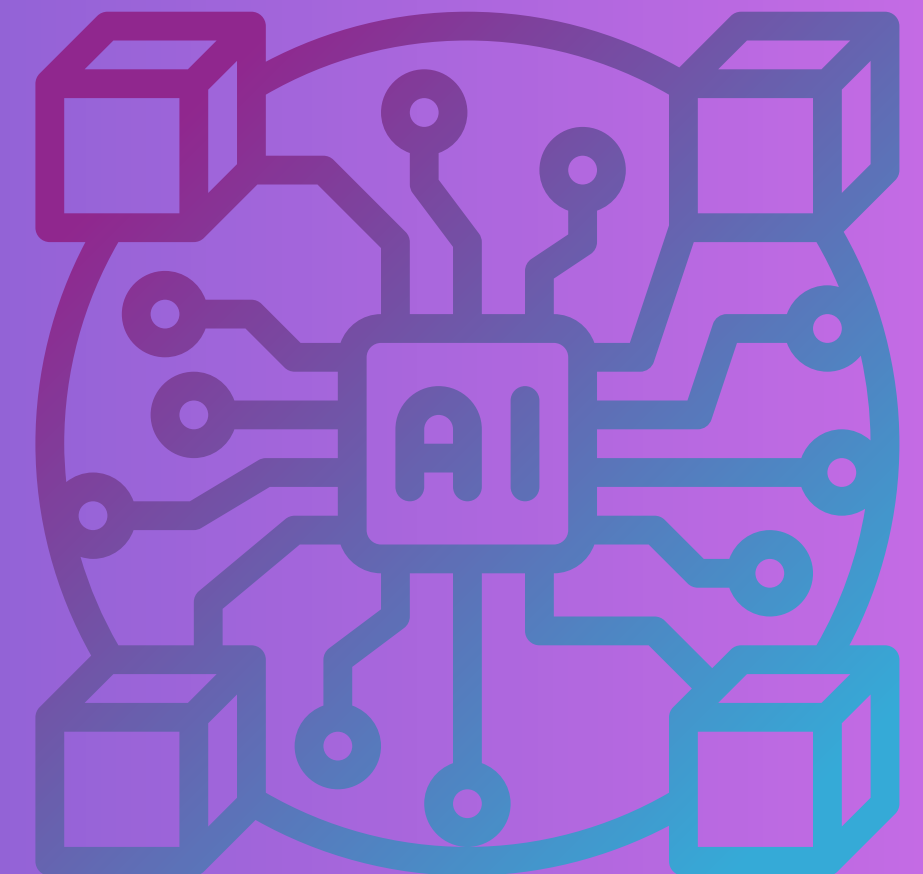
GPT-2 is well-known for its strong text generation capabilities. It can generate coherent and contextually relevant text across a wide range of topics and writing styles.

## Fine-tuning

GPT-2 can be fine-tuned on specific datasets or tasks to improve its performance on particular text generation tasks.

## Conditional Generation

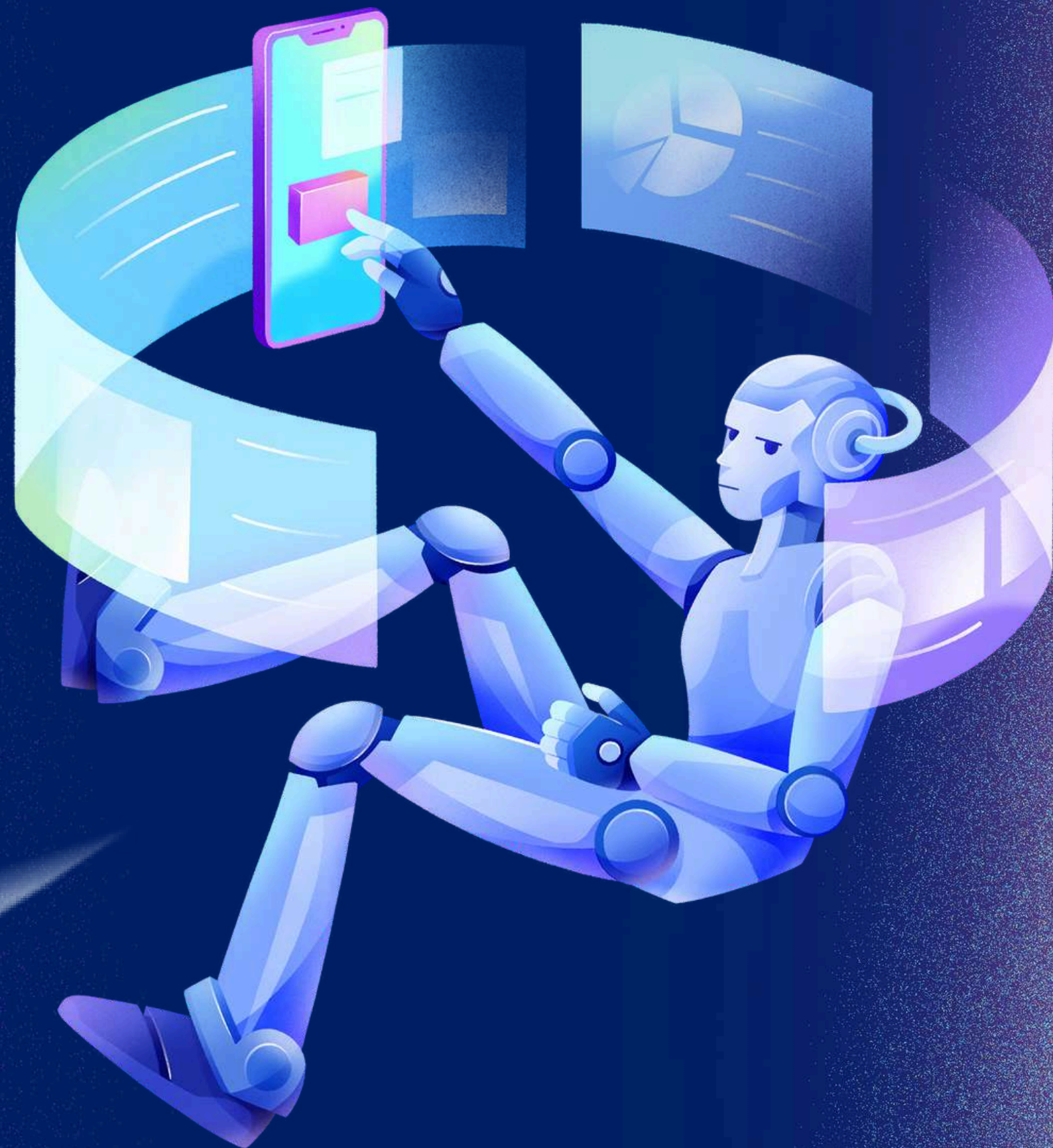
GPT-2 can generate text conditioned on prompts or input sequences, allowing users to control the content and style of the generated text.





# PARAMETERS

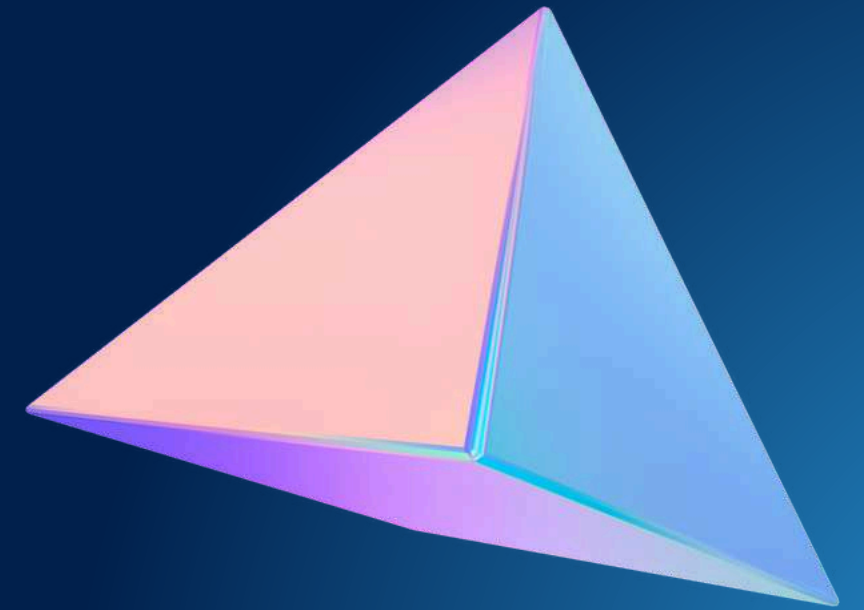
We used around 124 m parameter



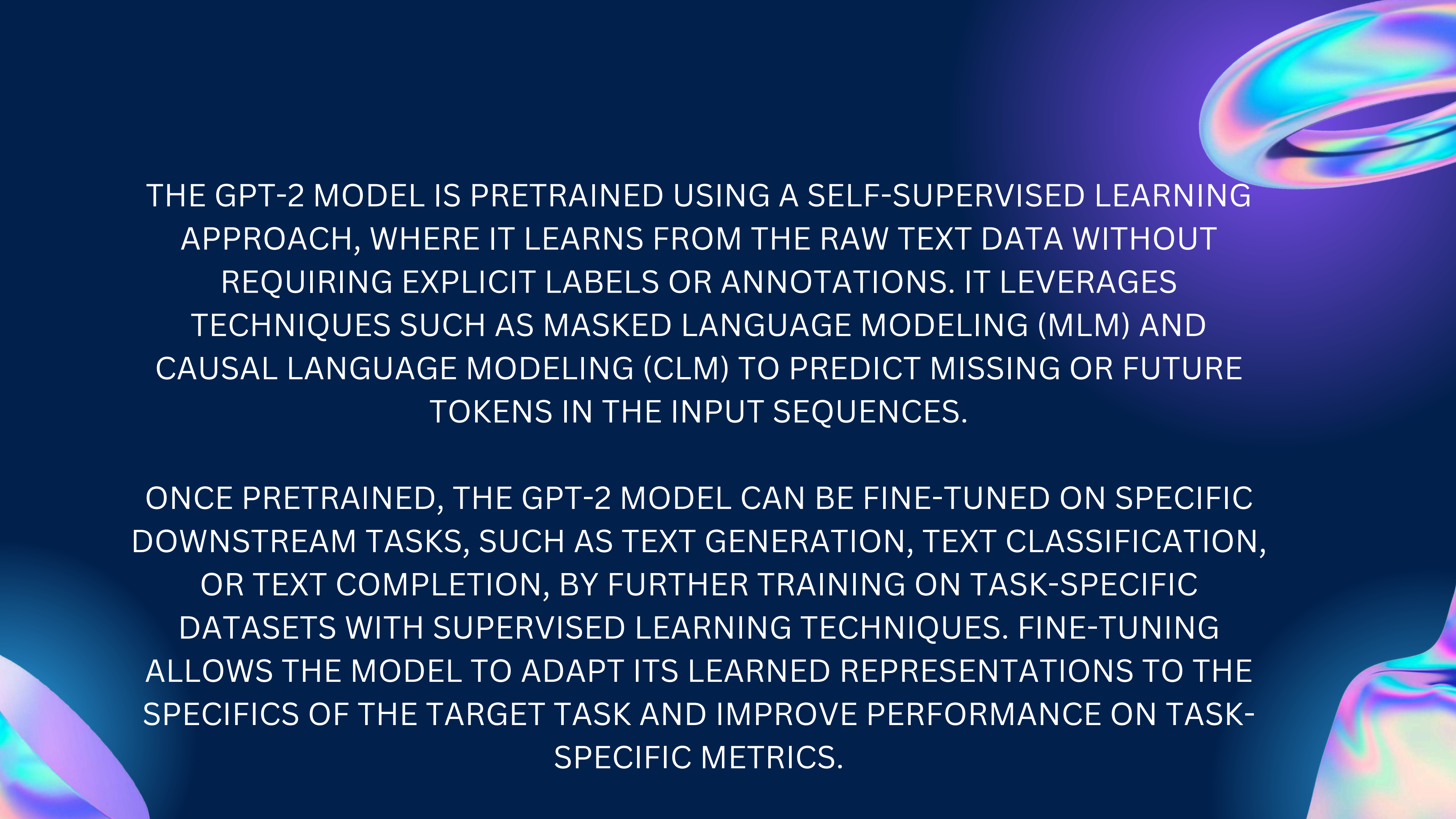


The GPT-2 model is pretrained using a large corpus of text data. Specifically, it is trained using a variant of the Transformer architecture called the "decoder-only" transformer, where only the decoder part of the Transformer is used. This architecture is well-suited for autoregressive language modeling tasks, where the model predicts the next token in a sequence given the previous tokens.

The pretraining objective of the GPT-2 model is to maximize the likelihood of predicting the next token in a sequence of text. During pretraining, the model is exposed to a large amount of text data, such as books, articles, websites, and other sources of text available on the internet. The model learns to generate coherent and contextually relevant text by capturing patterns and relationships in the data.







THE GPT-2 MODEL IS PRETRAINED USING A SELF-SUPERVISED LEARNING APPROACH, WHERE IT LEARNS FROM THE RAW TEXT DATA WITHOUT REQUIRING EXPLICIT LABELS OR ANNOTATIONS. IT LEVERAGES TECHNIQUES SUCH AS MASKED LANGUAGE MODELING (MLM) AND CAUSAL LANGUAGE MODELING (CLM) TO PREDICT MISSING OR FUTURE TOKENS IN THE INPUT SEQUENCES.

ONCE PRETRAINED, THE GPT-2 MODEL CAN BE FINE-TUNED ON SPECIFIC DOWNSTREAM TASKS, SUCH AS TEXT GENERATION, TEXT CLASSIFICATION, OR TEXT COMPLETION, BY FURTHER TRAINING ON TASK-SPECIFIC DATASETS WITH SUPERVISED LEARNING TECHNIQUES. FINE-TUNING ALLOWS THE MODEL TO ADAPT ITS LEARNED REPRESENTATIONS TO THE SPECIFICS OF THE TARGET TASK AND IMPROVE PERFORMANCE ON TASK-SPECIFIC METRICS.

# FINE TUNING

**Fine-tuning in text generation refers to the process of further training a pre-trained language model on a specific dataset or task to improve its performance on that particular task or dataset. In this context, "fine-tuning" means adjusting the parameters of the model slightly rather than training it from scratch.**



# FINE TUNING

**Fine-tuning is commonly used in natural language processing tasks such as text classification, sentiment analysis, language translation, and text generation to adapt pre-trained models to specific domains or tasks. It helps leverage the knowledge encoded in the pre-trained model while tailoring it to the specific requirements of the target task or dataset.**

# FINE TUNING

**For example, you can use the following command to create a new file in `/var/lib/python2.7/site-packages.py`. This file will be created in the root directory of your local Python installation. You can also specify the path to the file by passing it as the first argument, e.g. `/usr/local/bin/pip install -r requirements.txt`. You will need to add this line to end of sequence**



```
✓ [34] output = model.generate(input_ids,
14s                                max_length=max_length,
                                num_return_sequences=num_return_sequences, # Fixed parameter
                                num_beams=5, # Adjust for diversity
                                temperature=0.7, # Adjust for randomness
                                do_sample=True,
                                top_k=50, # Adjust for diversity
                                no_repeat_ngram_size=2
                                )
```

The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention\_mask` to the model. Setting `pad\_token\_id` to `eos\_token\_id`:50256 for open-end generation.

```
✓ 0s ▶ for i, sequence in enumerate(output):
      print(f"Generated sequence {i + 1}: {tokenizer.decode(sequence)}")
      print("\nEnd of sequence \n")
```

Generated sequence 1: python are very easy to use.

For example, you can use the following command to create a new file in /var/lib/python2.7/site-packages.py. This file will be created in the root of the environment.

End of sequence

Generated sequence 2: python are very easy to use.

For example, you can use the following command to create a new file in /var/lib/python2.7/site-packages.py. This file will be created in the root of the environment.

End of sequence

# BEFORE

# FINE

# TUNING



```
[29] model2 = GPT2LMHeadModel.from_pretrained("/content/output")
```

```
[30] output2 = model2.generate(input_ids,
                               max_length=max_length,
                               num_return_sequences=1,
                               num_beams=5, # Adjust for diversity
                               temperature=0.7, # Adjust for randomness
                               do_sample=True,
                               top_k=50, # Adjust for diversity
                               no_repeat_ngram_size=2,
                               pad_token_id=tokenizer.pad_token_id )
```

The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's `attention\_mask` to the model. Setting `pad\_token\_id` to `eos\_token\_id`:50256 for open-end generation.

```
▶ for i, sequence1 in enumerate(output2):
    print(f"Generated sequence {i + 1}: {tokenizer.decode(sequence1)}")
    print("\nEnd of sequence \n")
```

➞ Generated sequence 1: python are very similar to the python in terms of size and shape.  
"the python is a large python with a long neck and a thick tail that wraps around its body like a python."  
a python python?  
she looked down at her python, her eyes filled with tears as she stared at the dead body of her beloved python.  
he was alive, and his body was wrapped around her, his skin covered with scales and scales that looked like the skin of a giant python!

# AFTER

# FINE

# TUNING



THANK YOU!

