Transformers

Zeham Management Technologies BootCamp

by SDAIA

September the 11th, 2024

الهيئة السعودية للبيانات والذكاء الاصطناعي

Saudi Data & AI Authority

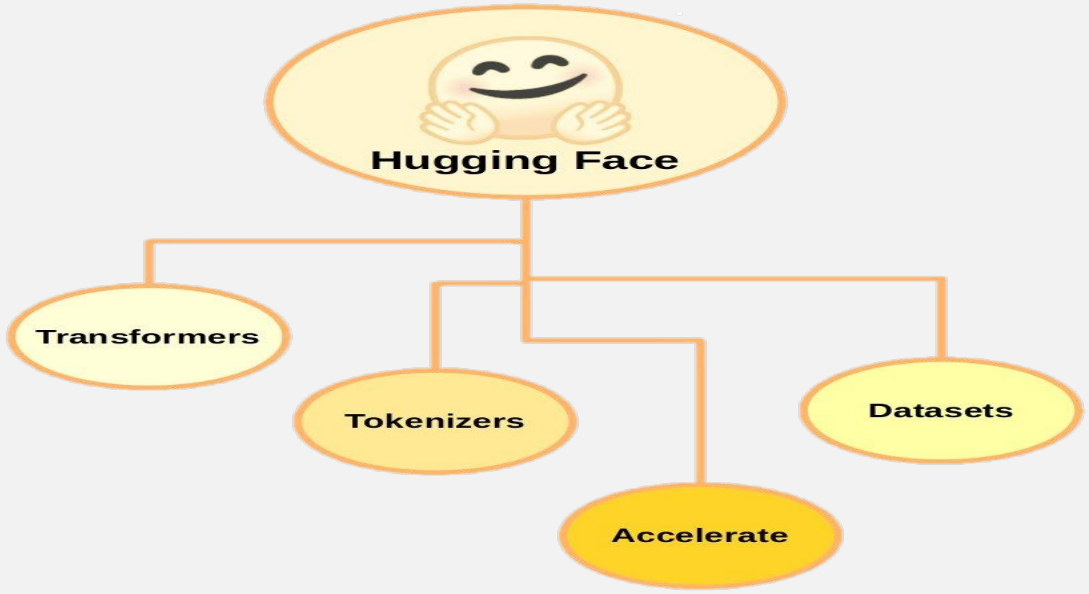
Agenda

- Hugging Face Ecosystem
- Pretrained Models

Hugging Face Ecosystem

What is Hugging Face?

Hugging Face Transformers is an open-source Python library that provides access to thousands of pre-trained Transformers models for natural language processing (NLP), computer vision, audio tasks, and more



Hugging Face Transformers

Tools:

- Transformers:

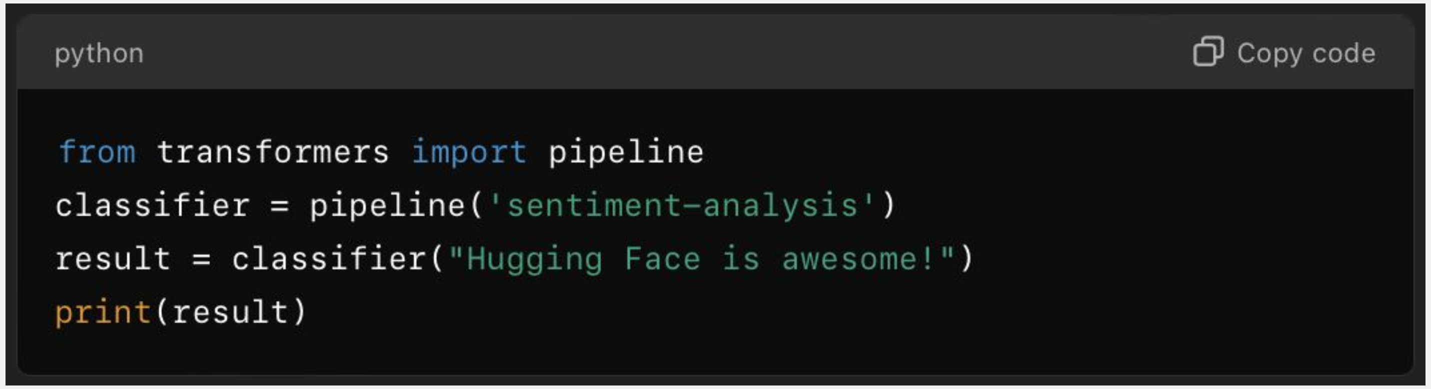
Load and use pretrained models.

- Datasets:

Access to standard datasets for various tasks.

- Hands-on:

Code snippet to load a model from the Hugging Face Hub.



```python
from transformers import pipeline
classifier = pipeline('sentiment-analysis')
result = classifier("Hugging Face is awesome!")
print(result)
```

Tokenization In Transformers

Tokenization is the process of converting text into smaller chunks (tokens) that can be used as inputs to machine learning models, including transformers.

Tokenization Techniques:

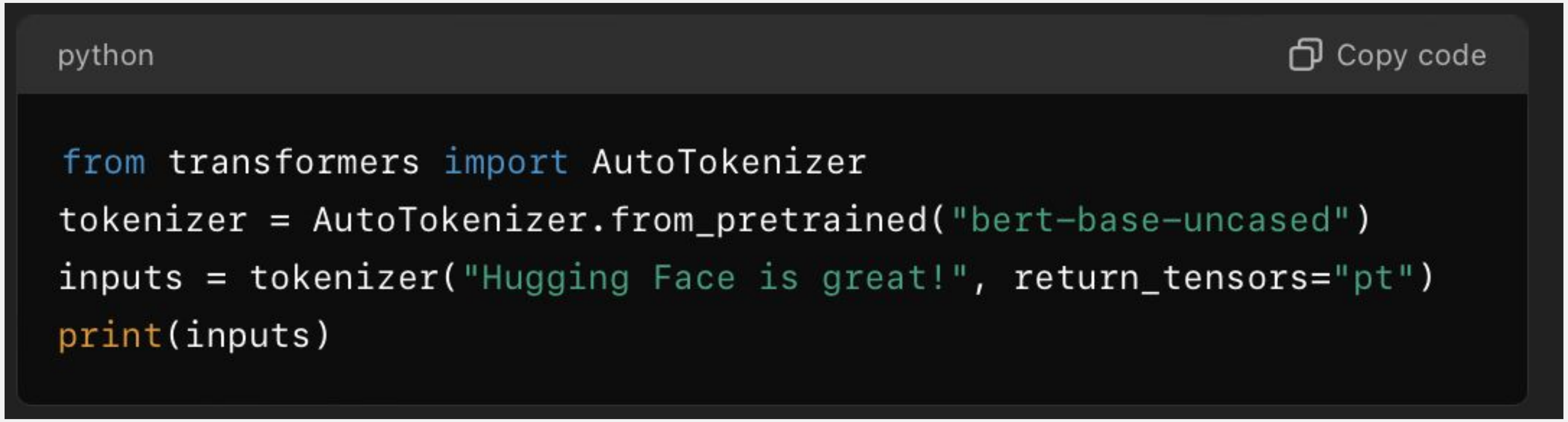
- Byte-Pair Encoding (BPE)
- WordPiece (used by BERT)
- SentencePiece (used by T5)

Transformers and other neural networks cannot directly process raw text (words or sentences) because they require numerical inputs.

Tokenization allows us to represent words and phrases in a format that models can understand, typically as numerical arrays (token IDs) that map to a vocabulary.

Tokenization In Transformers

- AutoTokenizer Example:

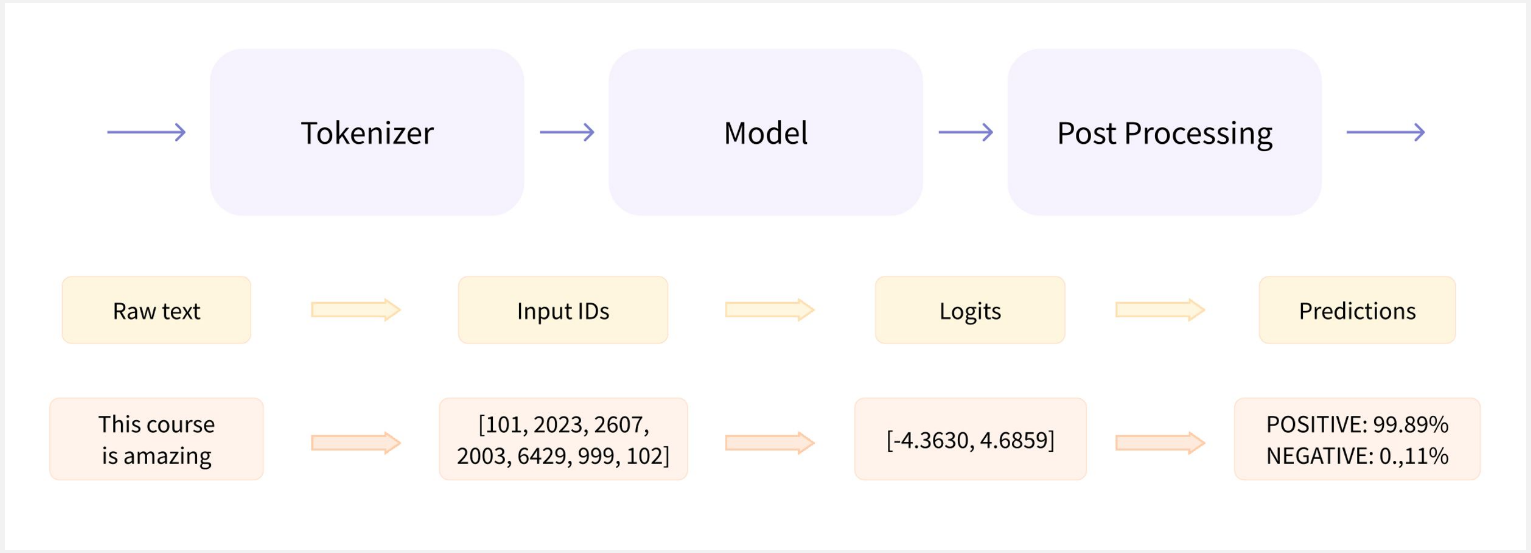


```python
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
inputs = tokenizer("Hugging Face is great!", return_tensors="pt")
print(inputs)
```

Pipeline

A pipeline is a high-level API provided by Hugging Face that simplifies the use of pretrained models for various common NLP tasks. It abstracts away the complexity of model loading, tokenization, and inference, allowing users to quickly get results without needing to delve into the technical details.

Behind the pipeline:



Pipeline

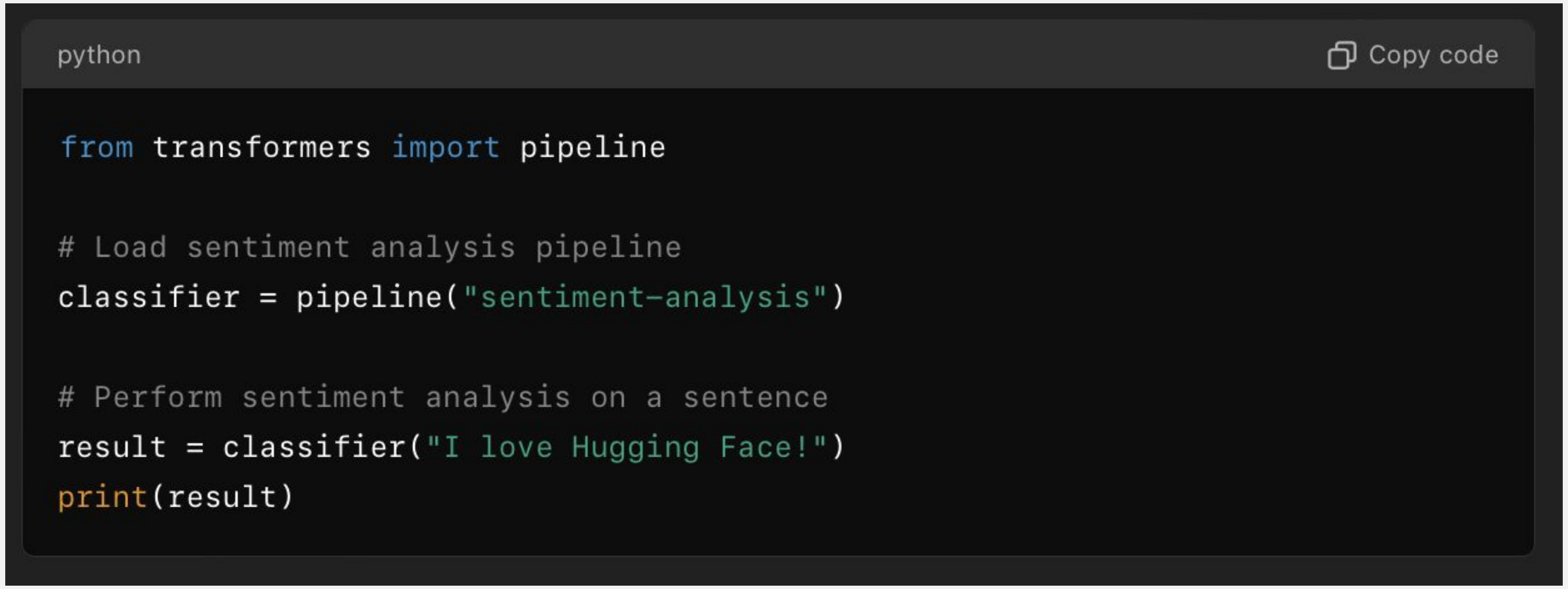
Hugging Face provides pipelines for many tasks, including:

- Text Classification: Classify text into categories (e.g., sentiment analysis).
- Named Entity Recognition (NER): Identify entities like people, places, organizations in text.
- Text Generation: Generate text based on a prompt (e.g., GPT models).
- Text Summarization: Summarize long texts.
- Translation: Translate text from one language to another.
- Fill-Mask: Fill in missing words in a sentence.

Pipeline Examples and Use Cases

- Sentiment Analysis Pipeline

To classify text as positive, negative, or neutral sentiment.

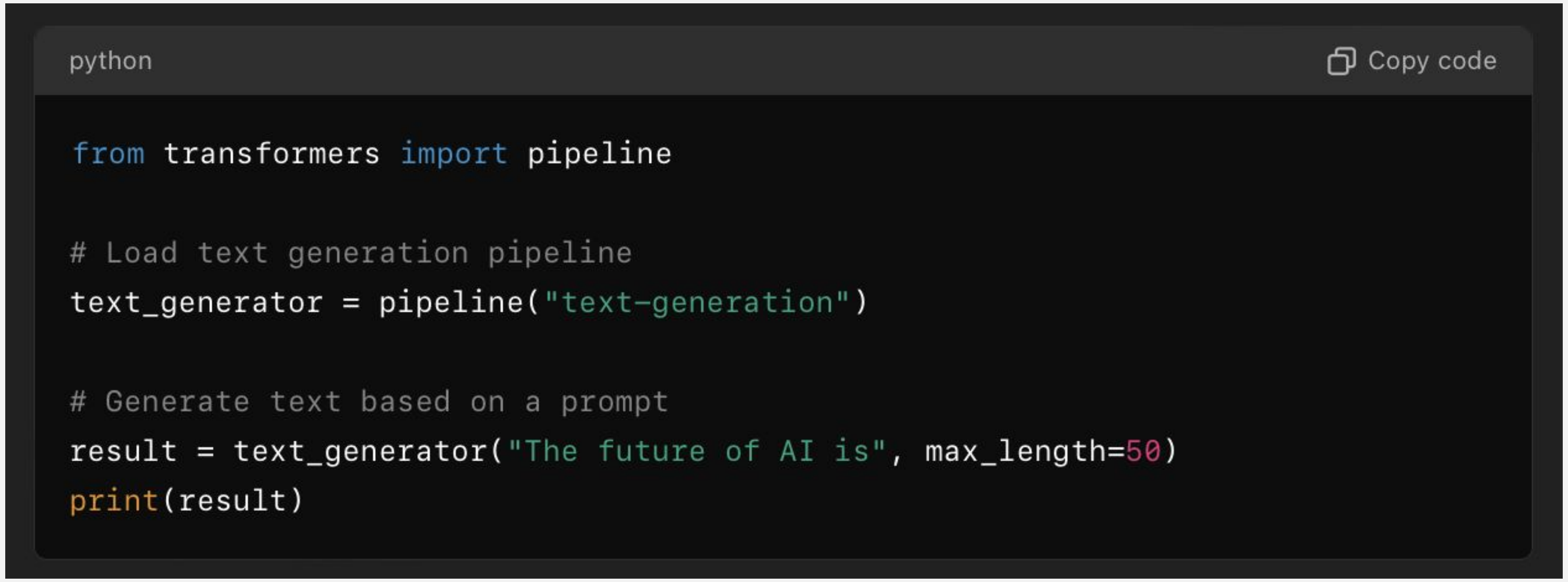


```python
from transformers import pipeline

# Load sentiment analysis pipeline
classifier = pipeline("sentiment-analysis")

# Perform sentiment analysis on a sentence
result = classifier("I love Hugging Face!")
print(result)
```

Pipeline Examples and Use Cases

- Text Generation Pipeline

To generate text from a given prompt, typically used for creative writing or auto-completion tasks.

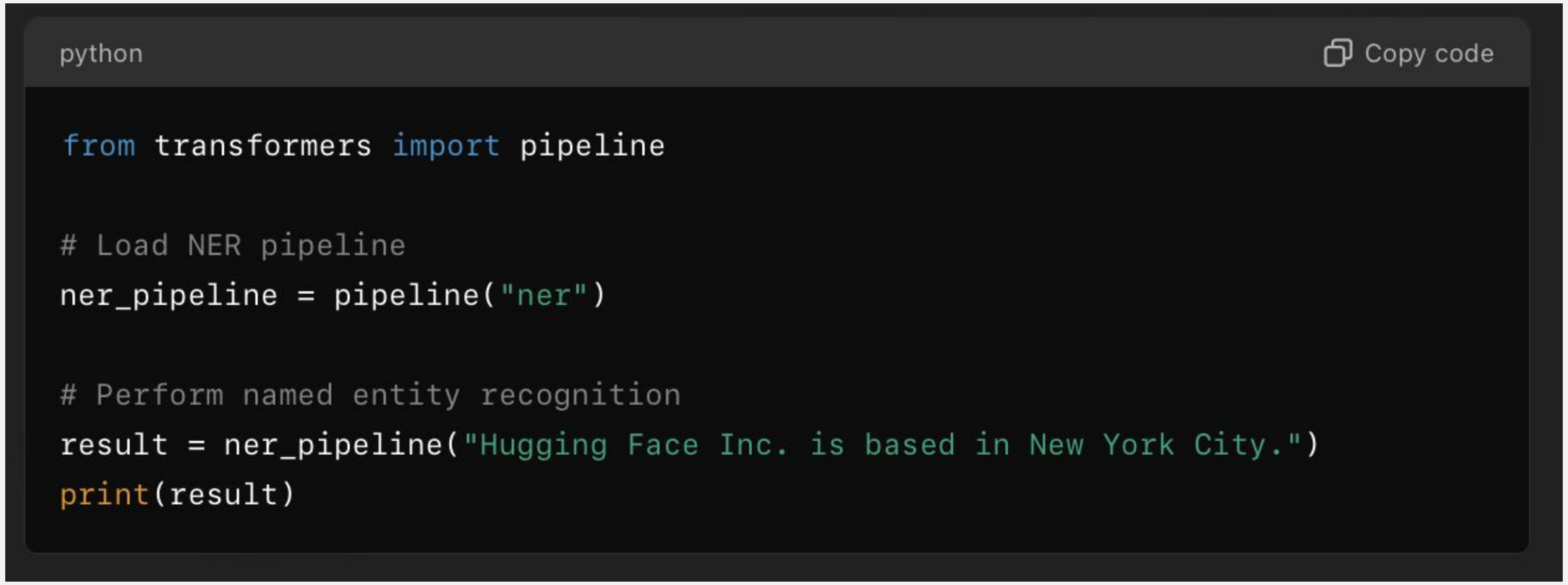


```python
from transformers import pipeline

# Load text generation pipeline
text_generator = pipeline("text-generation")

# Generate text based on a prompt
result = text_generator("The future of AI is", max_length=50)
print(result)
```

Pipeline Examples and Use Cases

- Named Entity Recognition (NER) Pipeline

To recognize named entities such as persons, organizations, locations, etc.

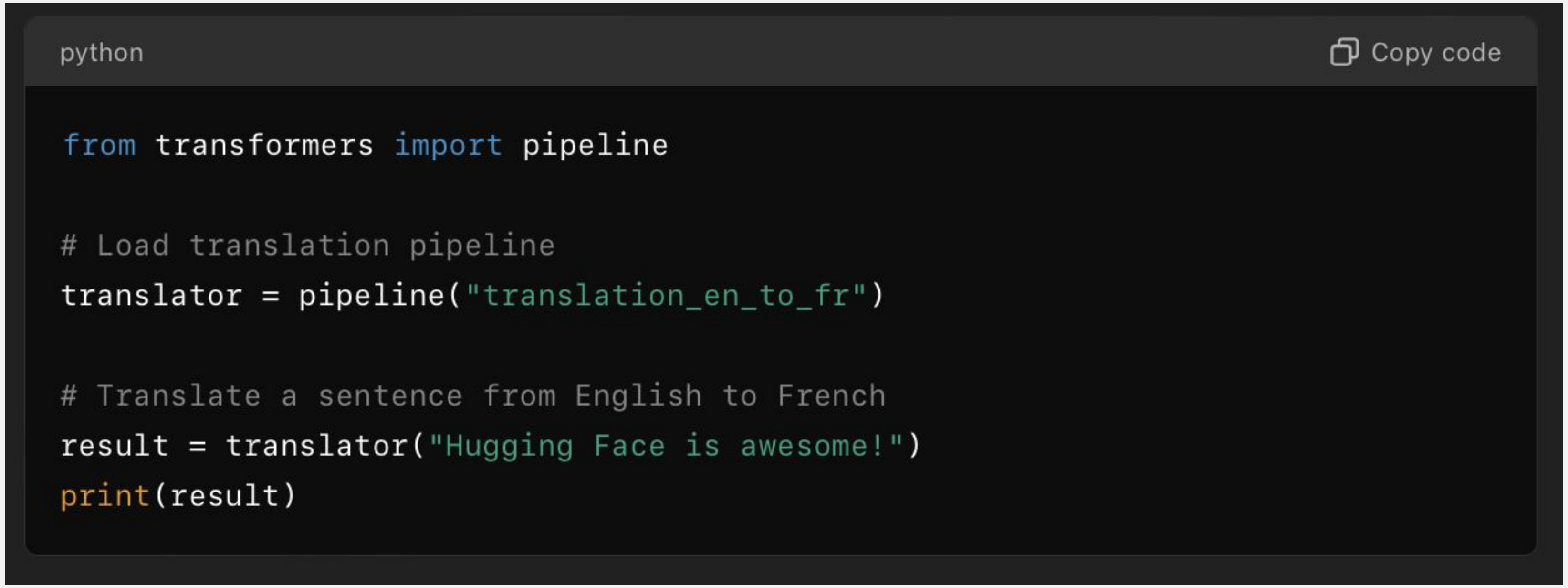


```python
from transformers import pipeline

# Load NER pipeline
ner_pipeline = pipeline("ner")

# Perform named entity recognition
result = ner_pipeline("Hugging Face Inc. is based in New York City.")
print(result)
```

Pipeline Examples and Use Cases

- Translation Pipeline

To translate text from one language to another.



```python
from transformers import pipeline

# Load translation pipeline
translator = pipeline("translation_en_to_fr")

# Translate a sentence from English to French
result = translator("Hugging Face is awesome!")
print(result)
```

Advantages Of Using Pipelines

- **Ease of Use:** With just a few lines of code, you can perform complex NLP tasks without worrying about the low-level implementation details.

- **Pretrained Models:** Hugging Face's pipelines come with access to state-of-the-art pretrained models like BERT, GPT-2, T5, RoBERTa, etc.

Advantages Of Using Pipelines

- **Task Flexibility:** You can switch between different tasks easily by just changing the pipeline name (e.g., from "text-generation" to "summarization").

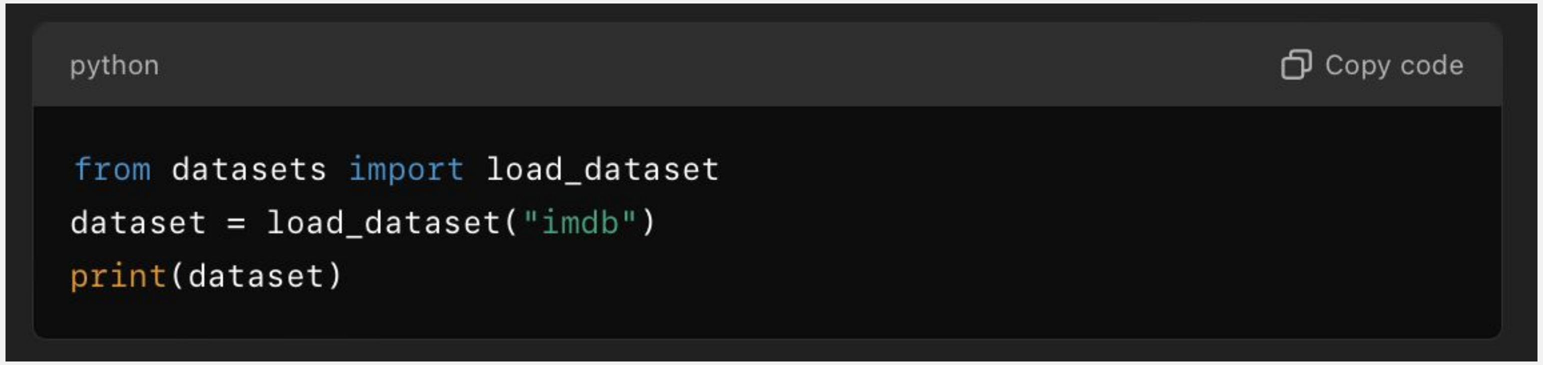
- **Customization:** Though pipelines are highly abstracted, you can still modify parameters such as model configuration, input settings (e.g., max_length), and output formats.

Loading Custom Datasets

- Fine-tuning models requires custom datasets for specialized tasks.

- Hugging Face datasets library simplifies loading and handling of large datasets.

Load a dataset using Hugging Face's datasets library:



```python
from datasets import load_dataset
dataset = load_dataset("imdb")
print(dataset)
```

Pre-Trained Models

Why Evaluate Pretrained Models?

Evaluating pretrained models is essential for:

- **Assessing Performance:** Determine how well a model performs on a specific task or dataset.
- **Comparing Models:** Compare the performance of different models to choose the one that best fits your needs.
- **Understanding Trade-offs:** Balancing between accuracy, speed, and memory usage depending on the application requirements.

Evaluate Pretrained Models

1. **Load the Model and Dataset**

Select the model and dataset that align with your use case.

2. **Perform Inference**

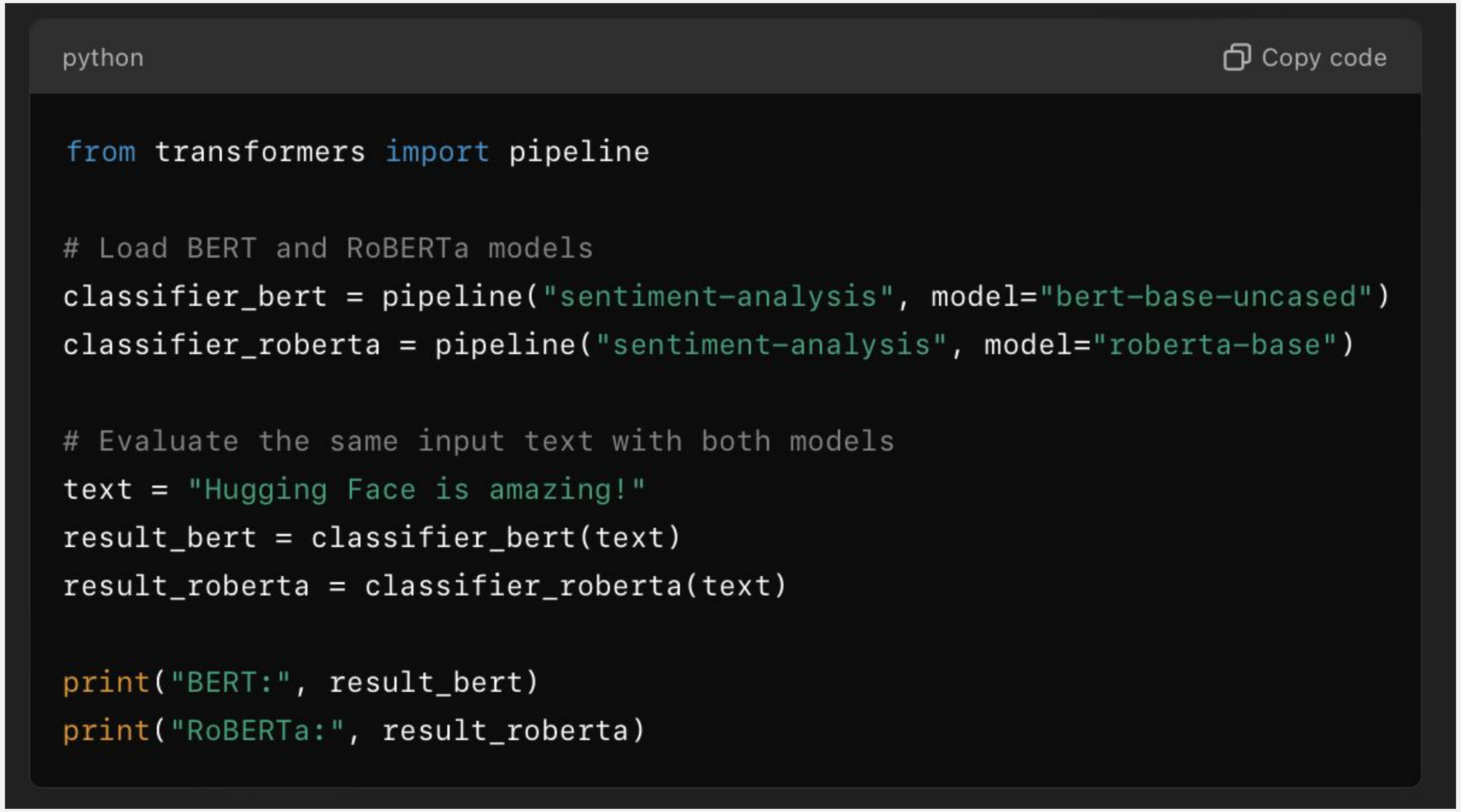
Run the pretrained model on a dataset to get predictions.

3. **Measure Accuracy and Other Metrics**

Compare the model’s predictions to the true labels to compute accuracy and other metrics.

Comparing Different Models

Example: Comparing BERT and RoBERTa for sentiment analysis.



```python
from transformers import pipeline

# Load BERT and RoBERTa models
classifier_bert = pipeline("sentiment-analysis", model="bert-base-uncased")
classifier_roberta = pipeline("sentiment-analysis", model="roberta-base")

# Evaluate the same input text with both models
text = "Hugging Face is amazing!"
result_bert = classifier_bert(text)
result_roberta = classifier_roberta(text)

print("BERT:", result_bert)
print("RoBERTa:", result_roberta)
```

Tutorial

8-Transformers/LAB/ Summarization.ipynb

Tutorial

8-Transformers/LAB/ Sentiment Analysis.ipynb

Thank you!

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority