Introducing large language models

INTRODUCTION TO LLMS IN PYTHON

Iván Palomares Carrascosa, PhD
Senior Data Science & AI Manager

datacamp

What are large language models (LLMs)?

What are large language models (LLMs)?

Generate new stories and articles

What are large language models (LLMs)?

Generate new stories and articles

Summarize a book

What are large language models (LLMs)?

Generate new stories and articles

Categorize a book

Summarize a book

What are large language models (LLMs)?

Generate new stories and articles

Categorize a book

Summarize a book

Translate

What are large language models (LLMs)?

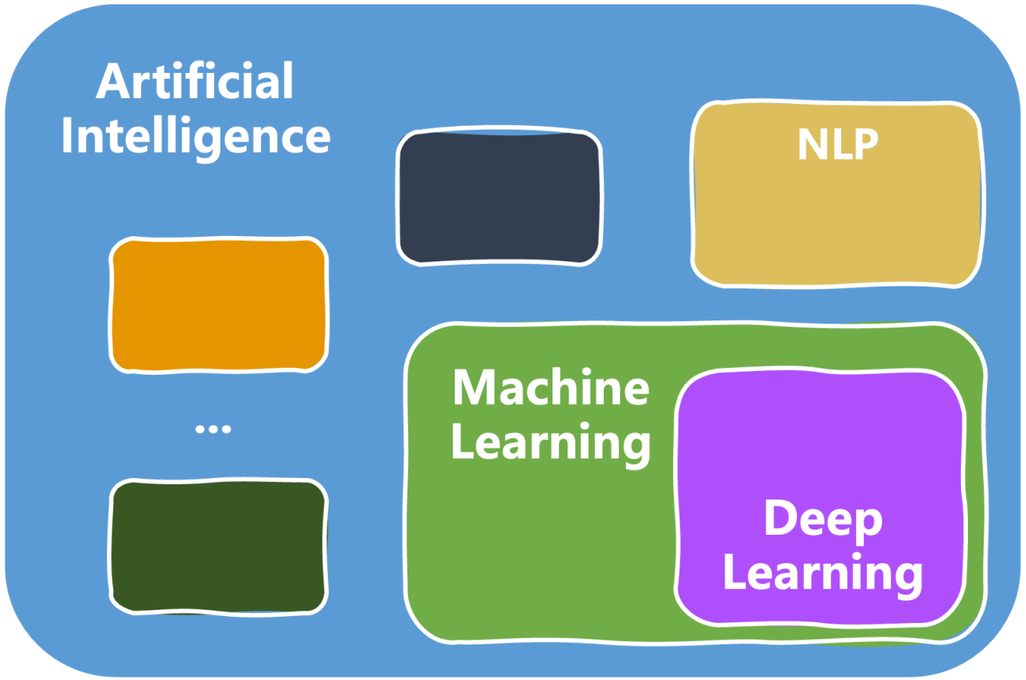
LLM: AI system able to **understand and generate human-like text** for addressing various complex **language tasks.**

- **Popular LLMs:** GPT, BERT, LLaMA, etc.

What are large language models (LLMs)?

Key characteristics of LLMs

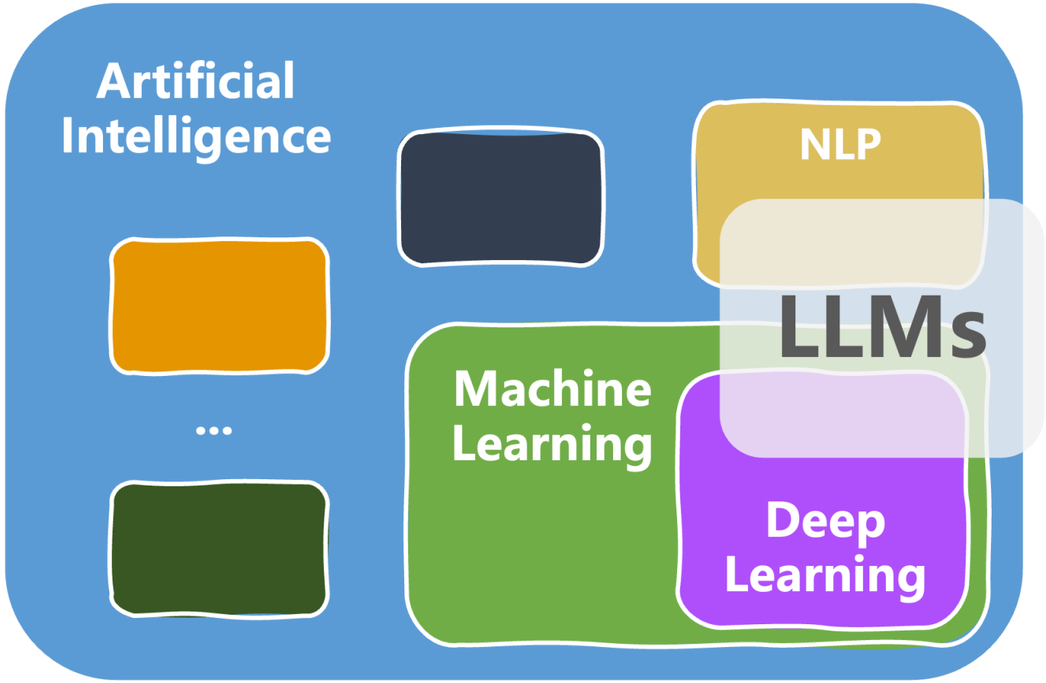
- Based on *deep learning* architectures, most commonly **transformers**.
  - Capture complex patterns in text data.
- Significant advances in **NLP tasks**:
  - Text generation, summarization, translation, question-answering, etc.



What are large language models (LLMs)?

Key characteristics of LLMs

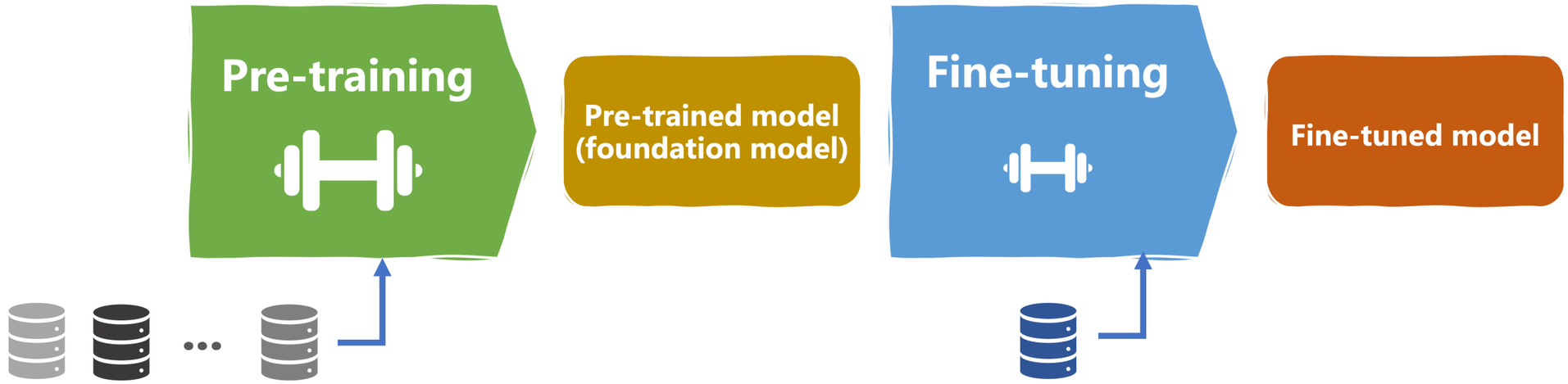
- Based on *deep learning* architectures, most commonly **transformers**.
  - Capture complex patterns in text data.
- Significant advances in **NLP tasks**:
  - Text generation, summarization, translation, question-answering, etc.
- Why "**large**"?
  - Very deep neural networks with lots of trainable weights.
  - Trained on vast text datasets.



LLMs development lifecycle

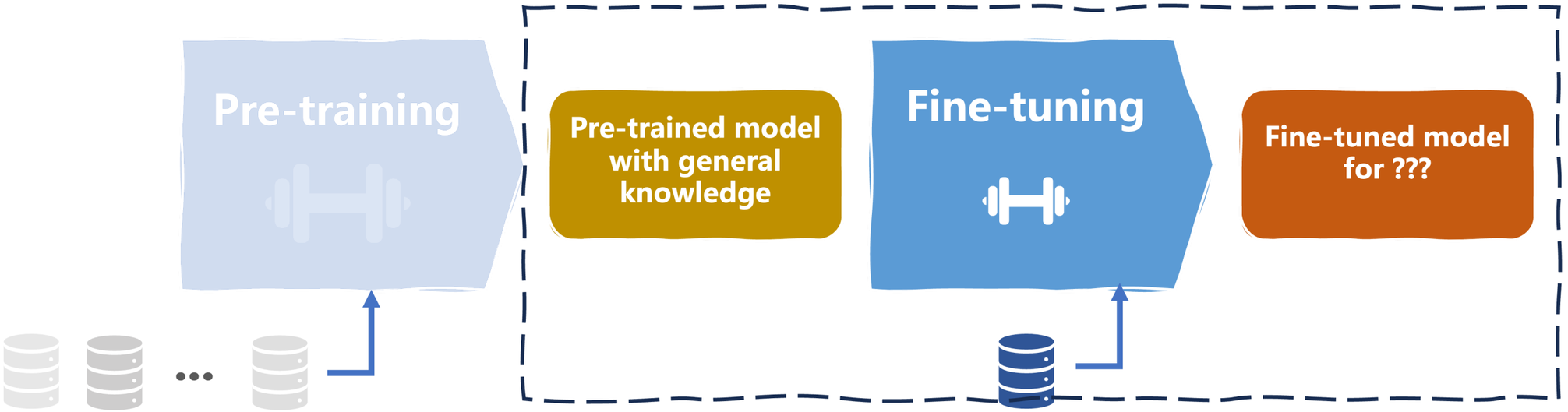
- Data ingestion and preparation.
- Model architecture design.

> **Pre-training and fine-tuning:**

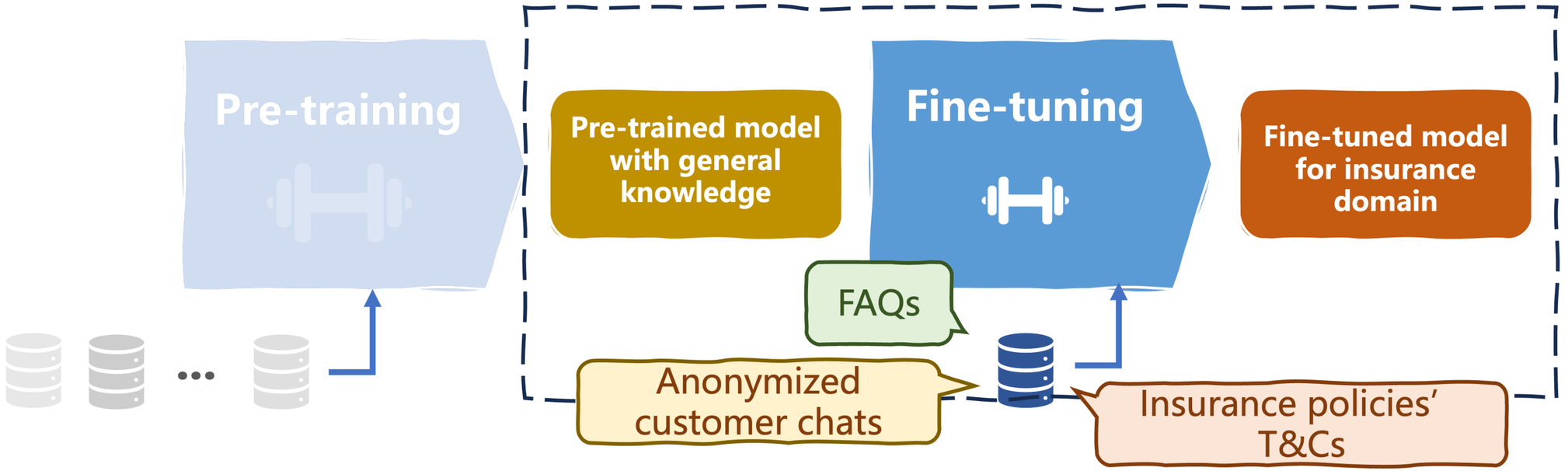


- Model evaluation, deployment, monitoring and maintenance.

Harnessing pre-trained LLMs: Hugging Face



Harnessing pre-trained LLMs: Hugging Face



Hugging Face : https://huggingface.co/

- A community hub of pre-trained LLMs and datasets ready to use.

"Hello World" example

- Load a sentiment classification model from **Hugging Face** hub, using **transformers library** and `pipeline()` function.
- Perform a simple sentiment prediction passing a customer review to the loaded LLM.

```
from transformers import pipeline
sentiment_classifier = pipeline("text-classification")
outputs = sentiment_classifier("""Dear seller, I got very impressed with the fast
                                  delivery and careful packaging of my order. Great
                                  experience overall, thank you!""")
print(outputs)
```

```
[{'label': 'POSITIVE', 'score': 0.9998602867126465}]
```

Let's practice!

INTRODUCTION TO LLMS IN PYTHON

Tasks LLMs can perform

INTRODUCTION TO LLMS IN PYTHON

Iván Palomares Carrascosa, PhD
Senior Data Science & AI Manager

datacamp

Language tasks: overview

Language Generation

Language Understanding

Language tasks: overview

Language Generation

- Text generation
- Code generation

Language Understanding

Language tasks: overview

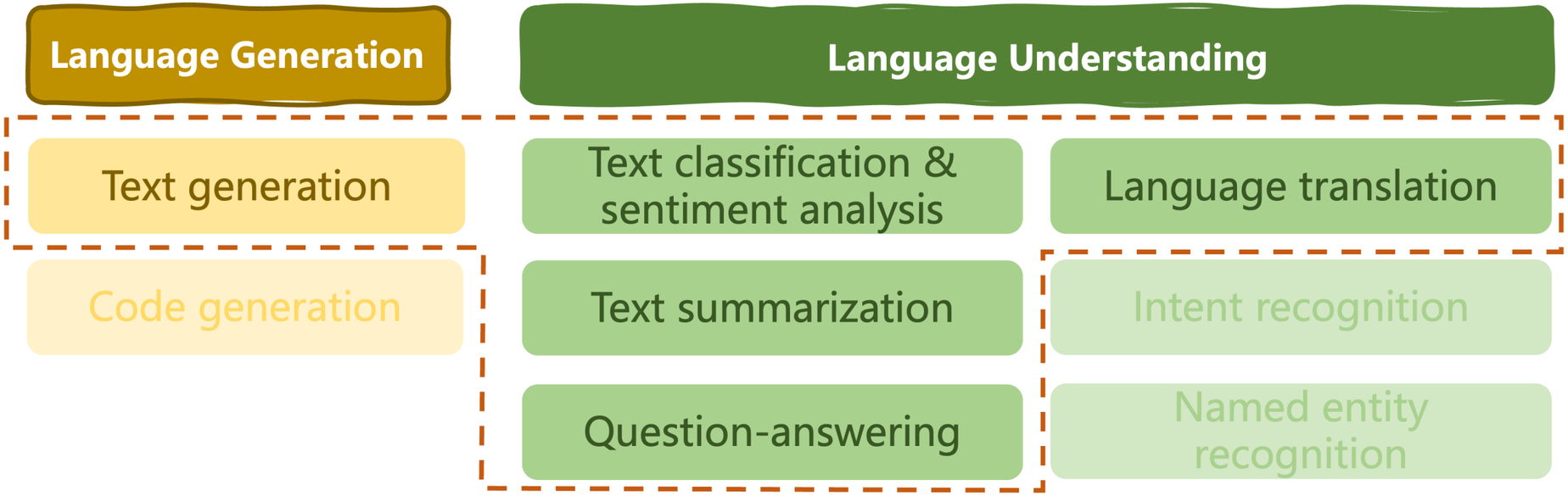
Language Generation

- Text generation
- Code generation

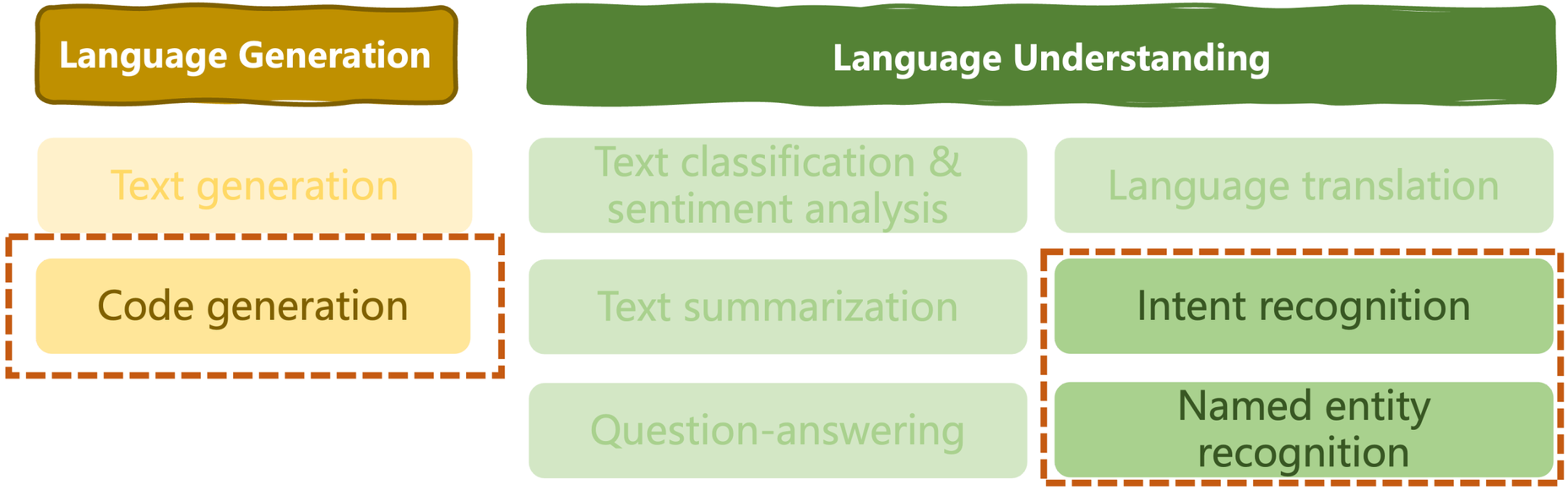
Language Understanding

- Text classification & sentiment analysis
- Language translation
- Text summarization
- Intent recognition
- Question-answering
- Named entity recognition

Language tasks: overview



Language tasks: overview



Language task	Goal
Code generation	Automatically generate code scripts based on requirements.
Intent recognition	Determine purpose behind a text, e.g., in chatbot enquiries.
Named entity recognition	Identify and classify named entities in text (people, places, ...).

Text classification: sentiment analysis


```
from transformers import pipeline

llm = pipeline("text-classification")
text = "Walking amid Gion's Machiya wooden houses was a mesmerizing experience"
outputs = llm(text)

print(outputs[0]['label'])
```

```
POSITIVE
```

Text generation


```
llm = pipeline("text-generation")
prompt = "The Gion neighborhood in Kyoto is famous for"
outputs = llm(prompt, max_length=100)
print(outputs[0]['generated_text'])
```

```
The Gion neighborhood in Kyoto is famous for making fish and seafood by the sea,
which made sense in the 1920s because it was the largest city of its age.
```

Text summarization


```
llm= pipeline("summarization", model="facebook/bart-large-cnn")
long_text = """Walking amid Gion's Machiya wooden houses is a mesmerizing experience. The beautifully
preserved structures exuded an old-world charm that transports visitors back in time, making them feel
like they had stepped into a living museum. The glow of lanterns lining the narrow streets add to the
enchanting ambiance, making each stroll a memorable journey through Japan's rich cultural history."""
outputs = llm(long_text, max_length=60, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

```
Walking amid Gion's Machiya wooden houses is a mesmerizing experience. The beautifully preserved
structures exuded an old-world charm. The glow of lanterns lining the narrow streets add to the ambiance.
Each stroll is an memorable journey through Japan's rich cultural history.
```

Question-answering


```
llm = pipeline("question-answering")
context = "Walking amid Gion's Machiya wooden houses was a mesmerizing experience."
question = "What are Machiya houses made of?"
outputs = llm(question=question, context=context)
print(outputs['answer'])
```

```
wooden
```

Language translation


```
llm = pipeline("translation_en_to_es", model="Helsinki-NLP/opus-mt-en-es")
text = "Walking amid Gion's Machiya wooden houses was a mesmerizing experience."
outputs = llm(text, clean_up_tokenization_spaces=True)
print(outputs[0]['translation_text'])
```

```
Caminar entre las casas de madera Machiya de Gion fue una experiencia fascinante.
```

Let's practice!

INTRODUCTION TO LLMS IN PYTHON

The transformer architecture

INTRODUCTION TO LLMS IN PYTHON

Iván Palomares Carrascosa, PhD
Senior Data Science & AI Manager

datacamp

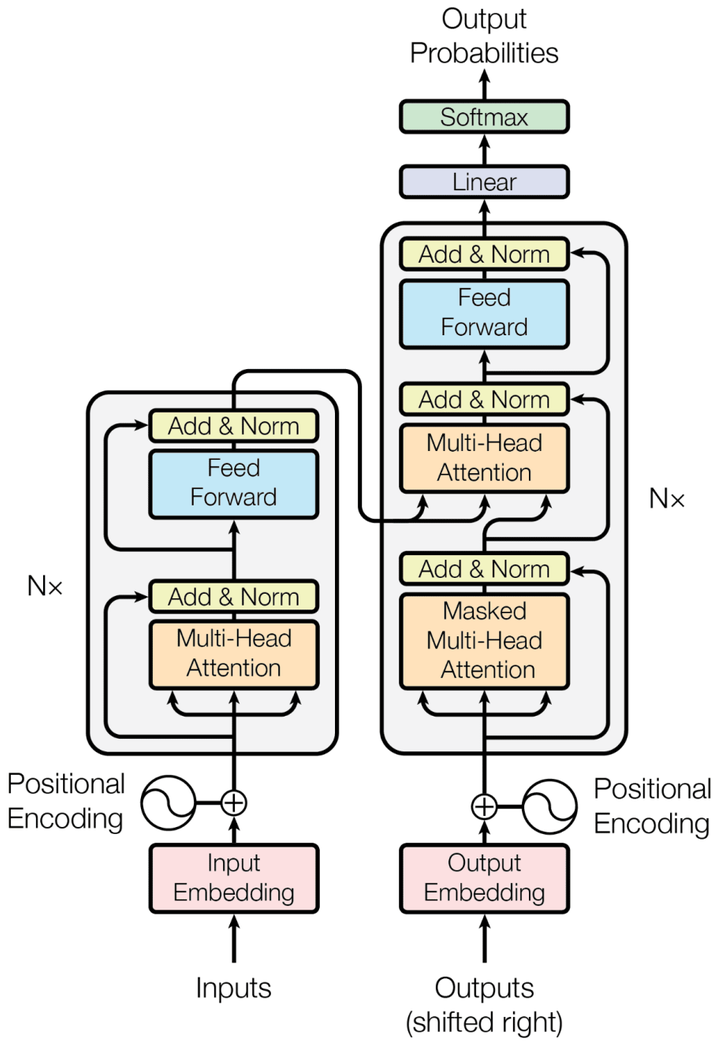
What is a transformer?

- **Deep learning** architecture for **text processing, understanding**, and **generation.**
- Foundation behind popular LLMs: **BERT, GPT, T5**, etc.

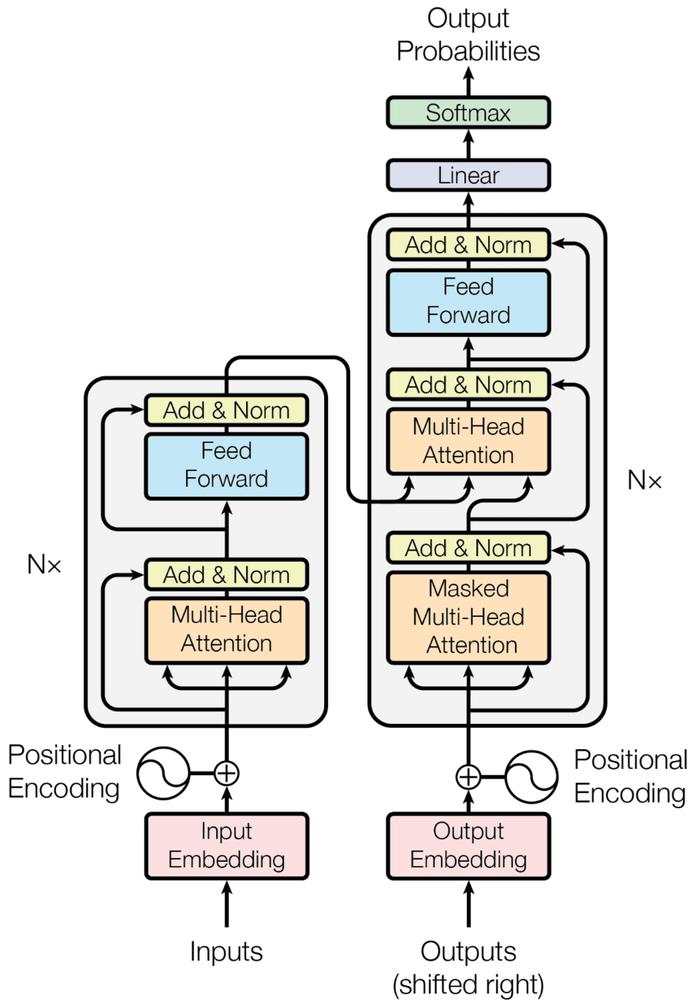
Characteristics:

- No recurrent (RNN) architecture.
- Capture **long-range dependencies** in text.
  - **Attention mechanisms + positional encoding.**
- Tokens are handled simultaneously
  - **Efficient:** faster training and inference.

The original transformer architecture



The original transformer architecture



- Two main stacks: **encoder** and **decoder.**
- **N encoder** (resp. decoder) **layers.**
  - Attention mechanisms + feed-forward.
- No **recurrence** nor **convolutions.**
- Intended for various **language tasks**, e.g.:
  - Translation
  - Summarization
  - Question-answering
  - etc.

[1] Image source: A. Vaswani, et al. "Attention is all you need". Arxiv, 2017: https://arxiv.org/pdf/1706.03762.pdf

Our first PyTorch transformer

Structural elements in a transformer architecture:

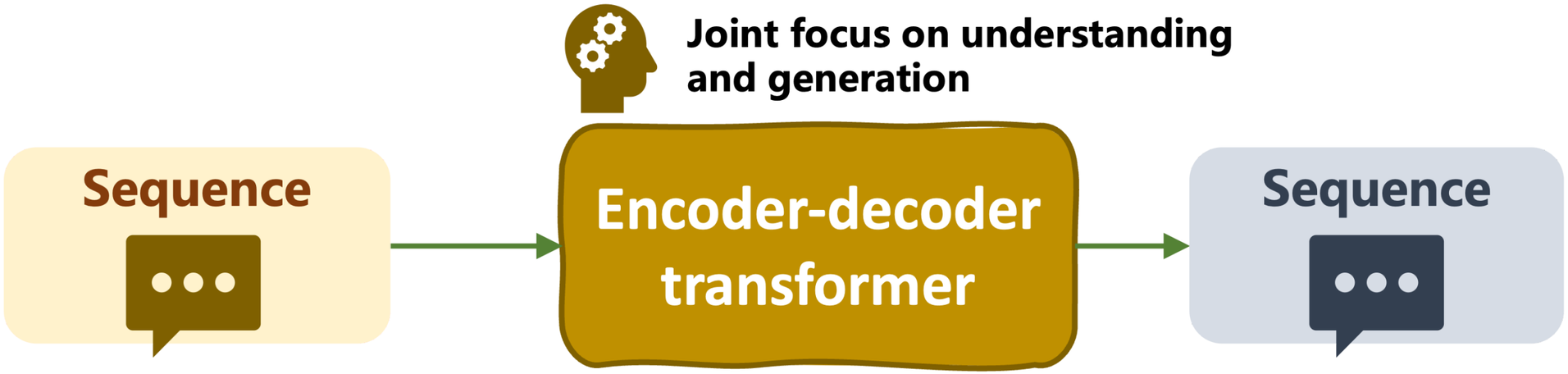
- `d_model` : model embedding dimension.
- `n_heads` : number of attention heads.
- `num_encoder_layers,num_decoder_layers` : number of encoder and decoder layers.

PyTorch transformer initialization.

- `torch.nn.Transformer` class.

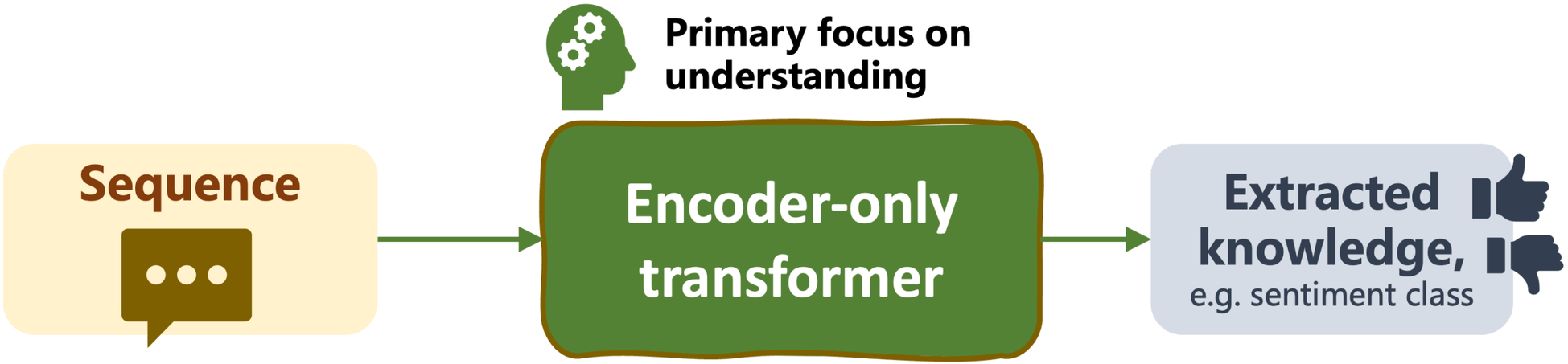
```
import torch
import torch.nn as nn

d_model = 512
n_heads = 8
num_encoder_layers = 6
num_decoder_layers = 6

model = nn.Transformer(
    d_model=d_model,
    nhead=n_heads,
    num_encoder_layers=num_encoder_layers,
    num_decoder_layers=num_decoder_layers
)
```

Types of transformer architectures



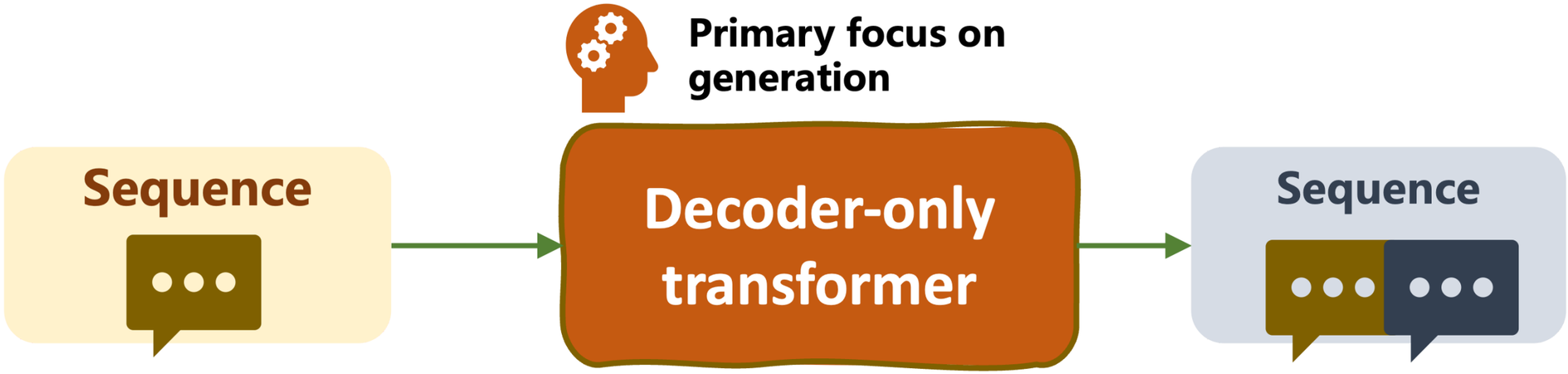
Type	Language tasks	Notable models
Encoder-decoder	Language translation, text summarization	T5, BART
-		
-		

Types of transformer architectures



Type	Language tasks	Notable models
Encoder-decoder	Language translation, text summarization	T5, BART
Encoder-only	**Text classification, extractive QA**	**BERT**
-		

Types of transformer architectures



Type	Language tasks	Notable models
Encoder-decoder	Language translation, text summarization	T5, BART
Encoder-only	Text classification, extractive QA	BERT
Decoder-only	**Text generation, generative QA.**	**GPT**

Let's practice!

INTRODUCTION TO LLMS IN PYTHON