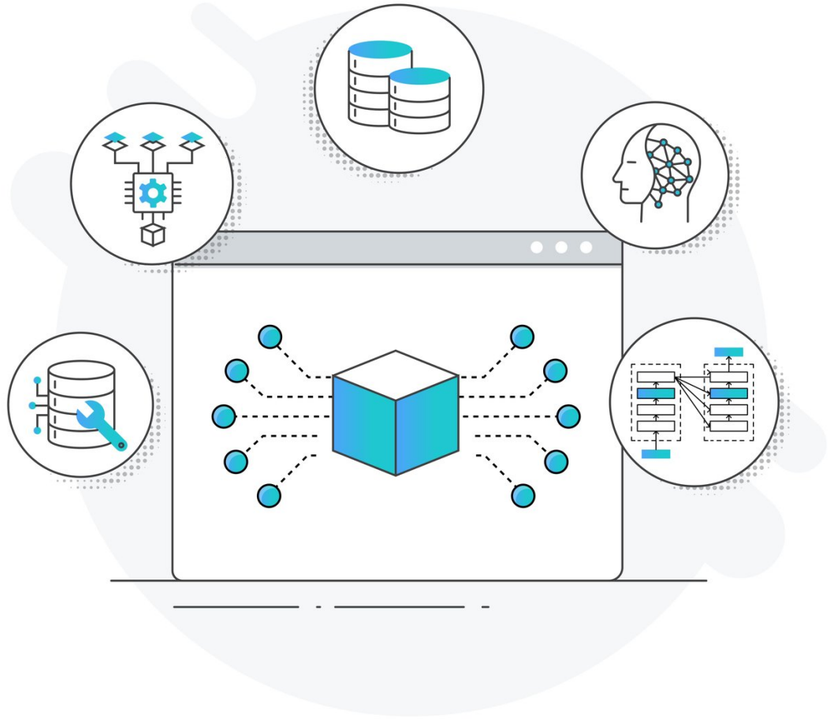
Introducing large language models

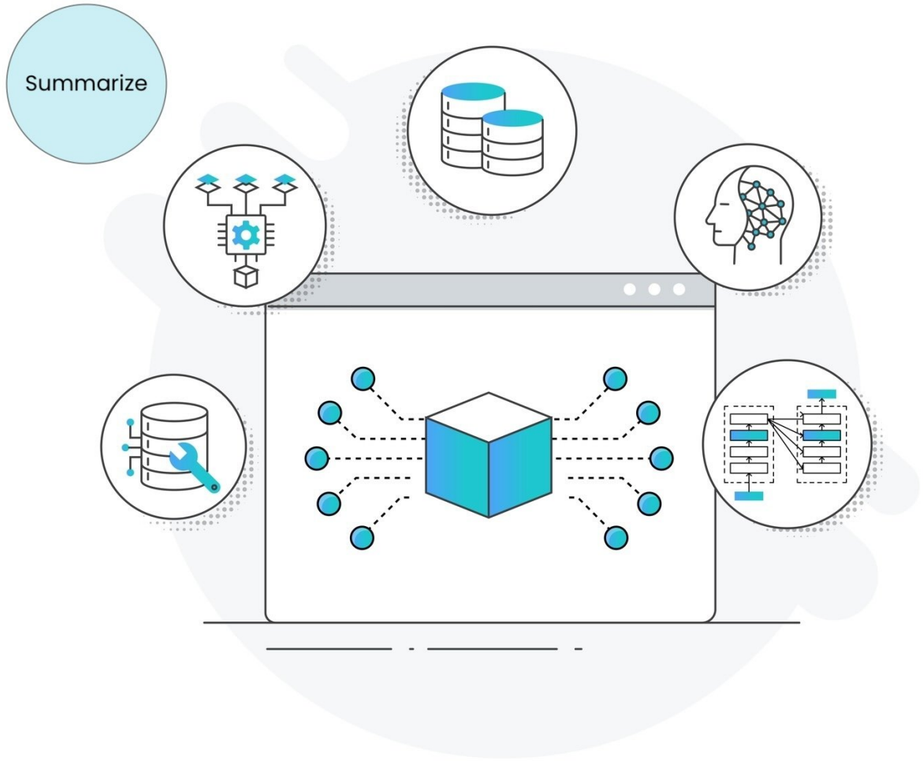
INTRODUCTION TO LLMS IN PYTHON

Iván Palomares Carrascosa, PhD
Senior Data Science & AI Manager

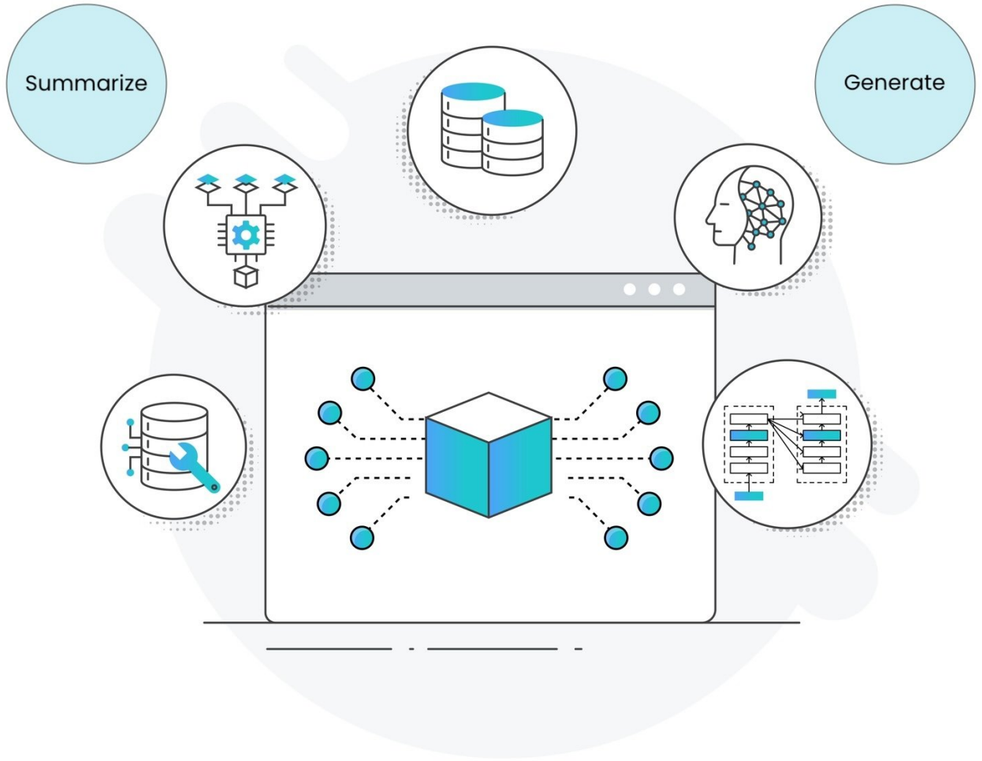
datacamp



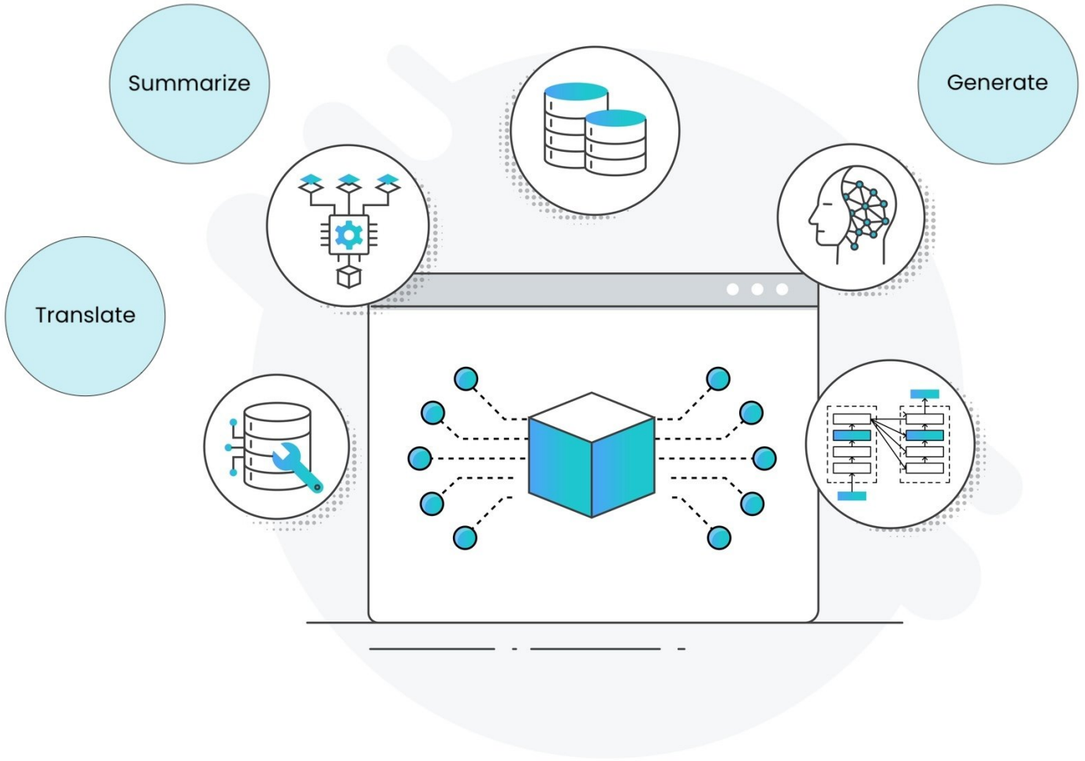
Large Language Models **(LLM)**



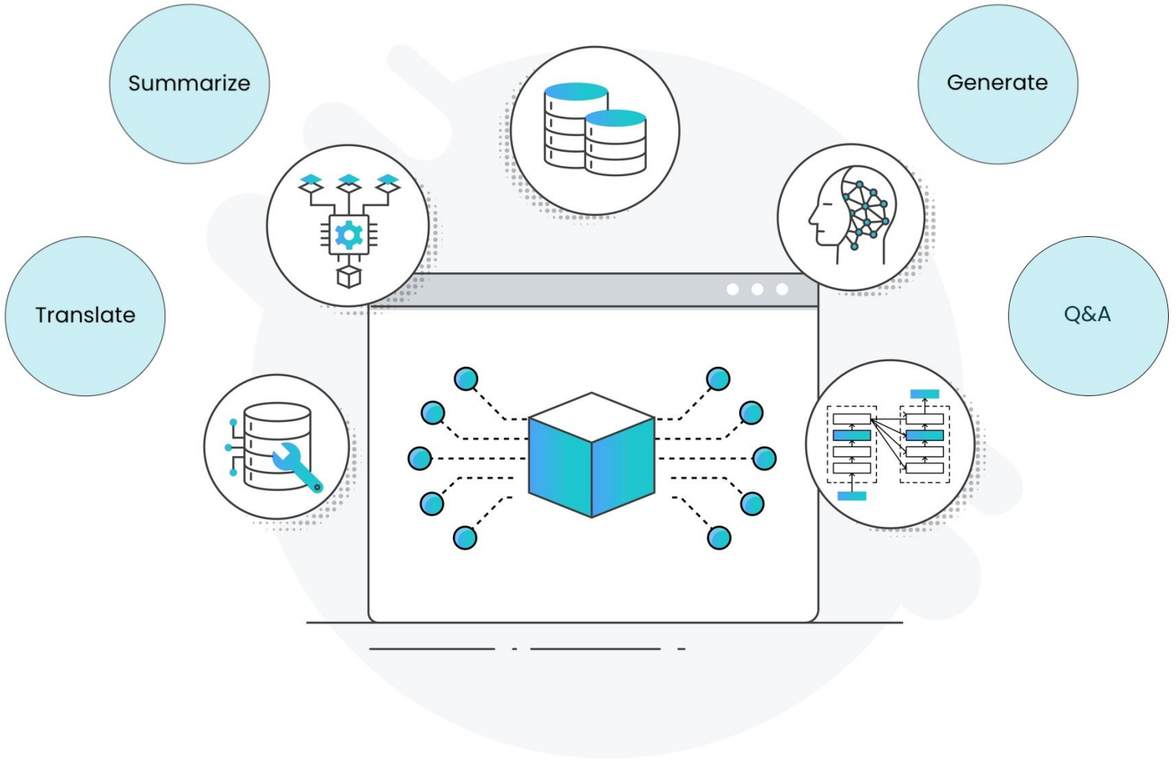
Large Language Models **(LLM)**



Large Language Models **(LLM)**



Large Language Models **(LLM)**



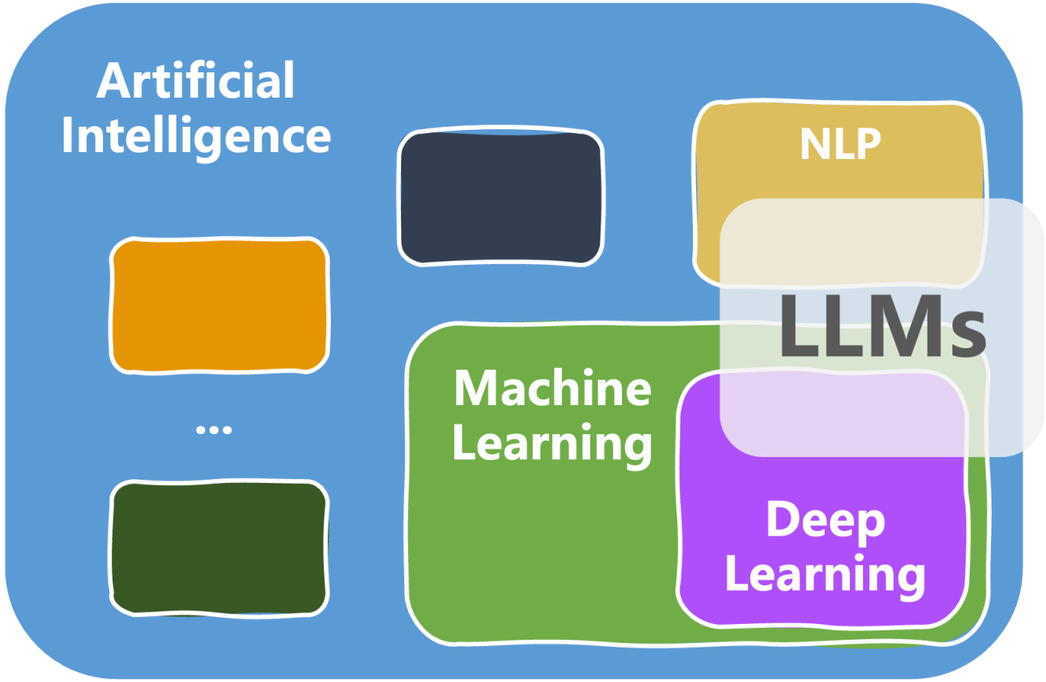
Large Language Models **(LLM)**

LLaMA by Meta

Gemini

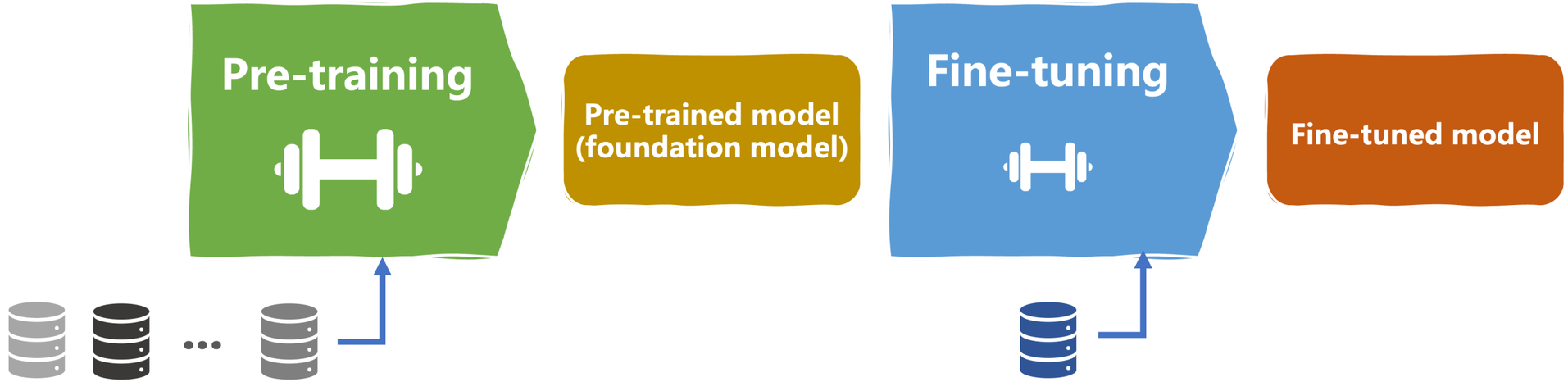
LLMs

- Based on **deep learning** architectures
- Most commonly **transformers**
- Capture complex patterns in text data
- Significant advances in **Natural Language Processing** (NLP) tasks
- Huge **neural networks** with lots of parameters
- Trained on enormous text datasets



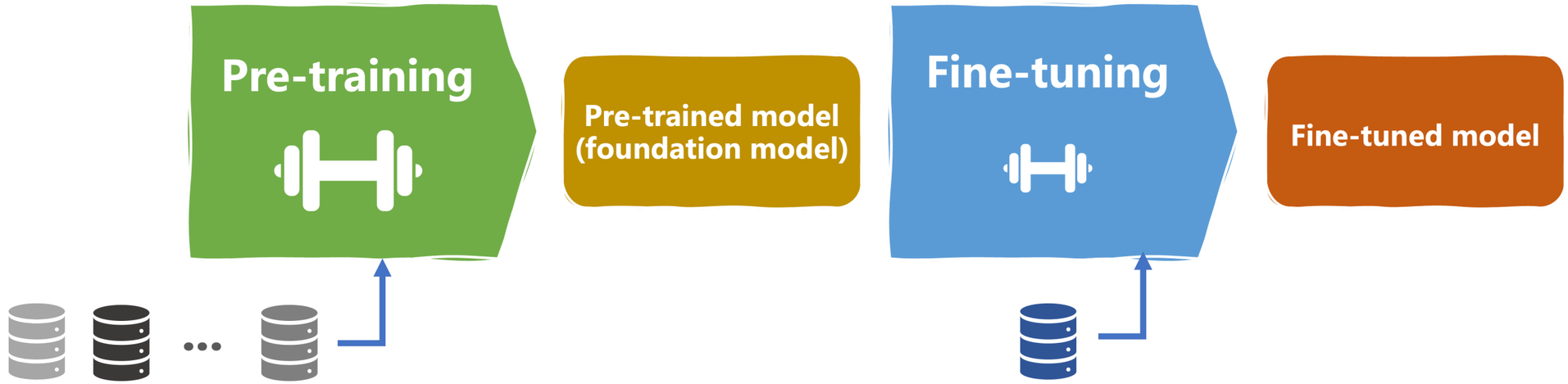
LLMs development lifecycle

- Data ingestion and preparation
- Model architecture design
- **Pre-training and fine-tuning:**



- Model evaluation, deployment, monitoring and maintenance

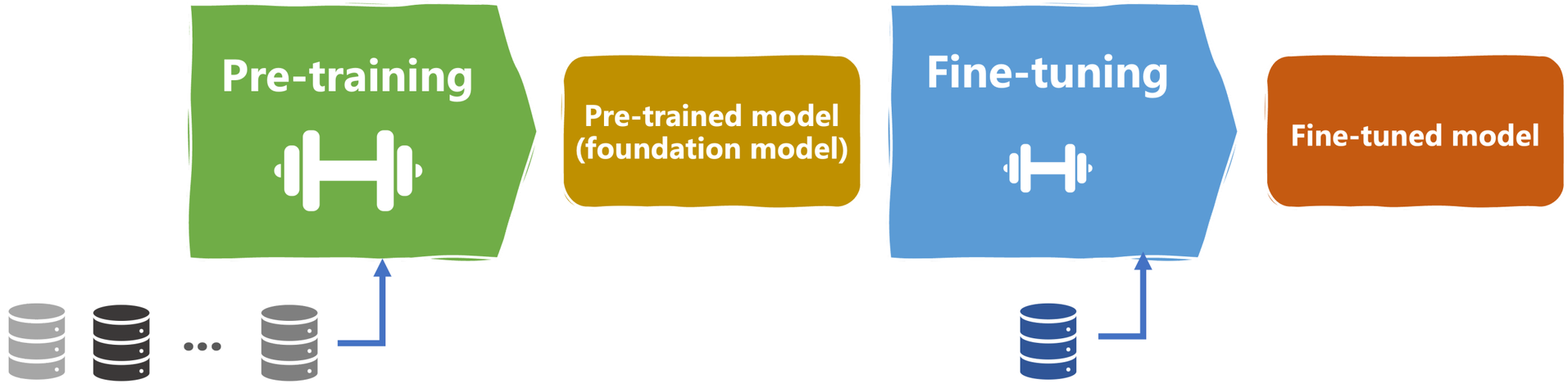
Pre-training and fine-tuning



Pre-training

- Learns patterns
- Computationally intensive
- Pre-trained foundation model

Pre-training and fine-tuning



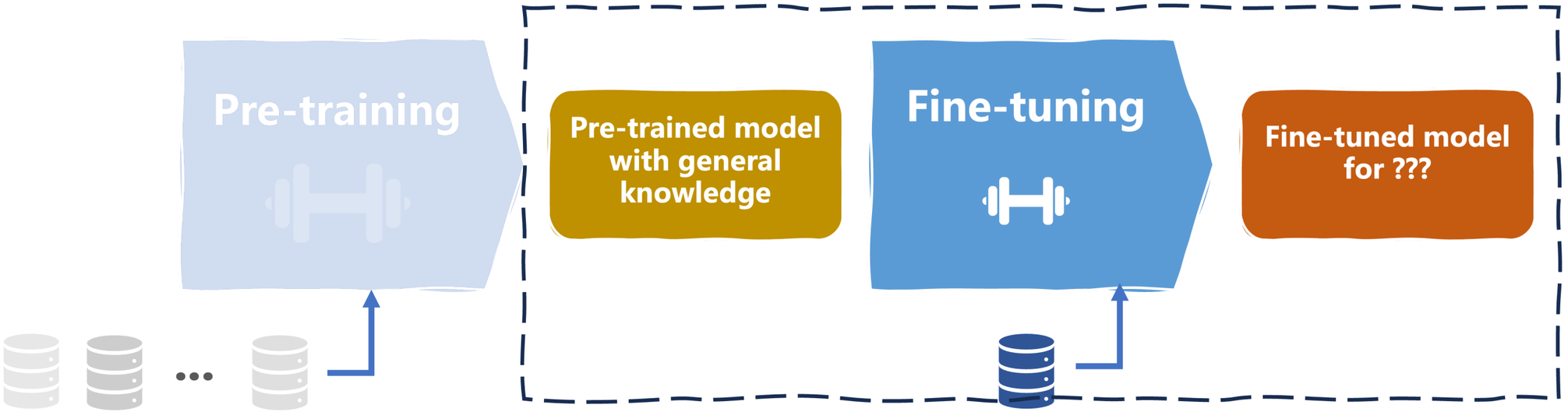
Pre-training

- Learns patterns
- Computationally intensive
- Pre-trained foundation model

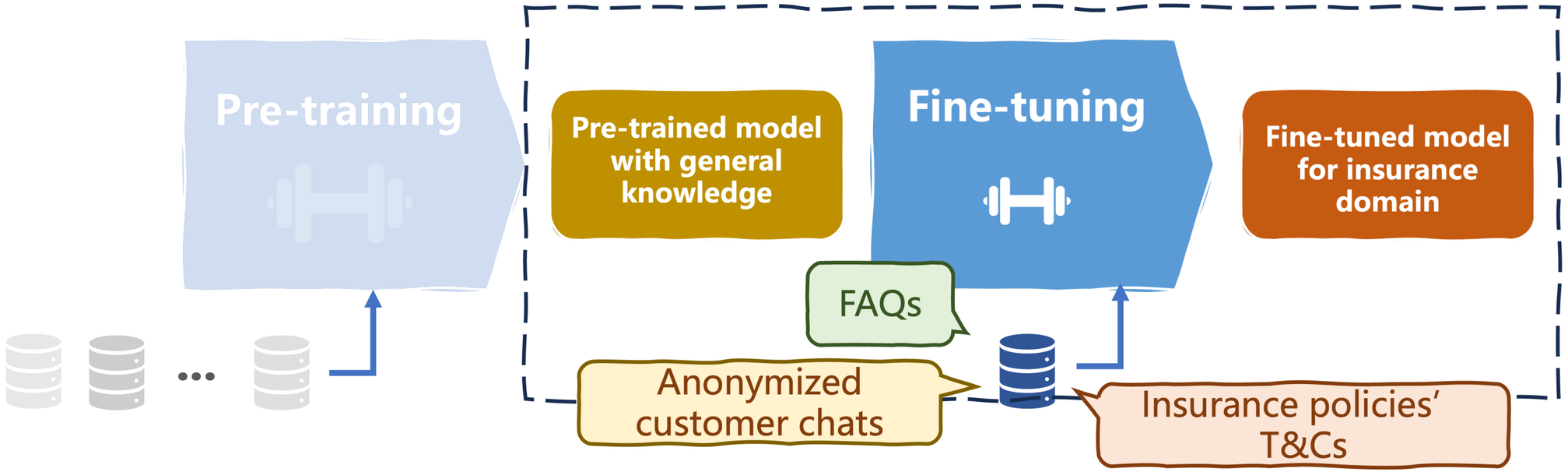
Fine-tuning

Domain specific

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

Using Hugging Face models

```
from transformers import pipeline

text_classifier = pipeline(task="text-classification",
                           model="nlptown/bert-base-multilingual-uncased-sentiment")

text = "Dear seller, I got very impressed with the fast delivery and careful packaging

sentiment = text_classifier(text)

print(sentiment)
```

```
[{'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

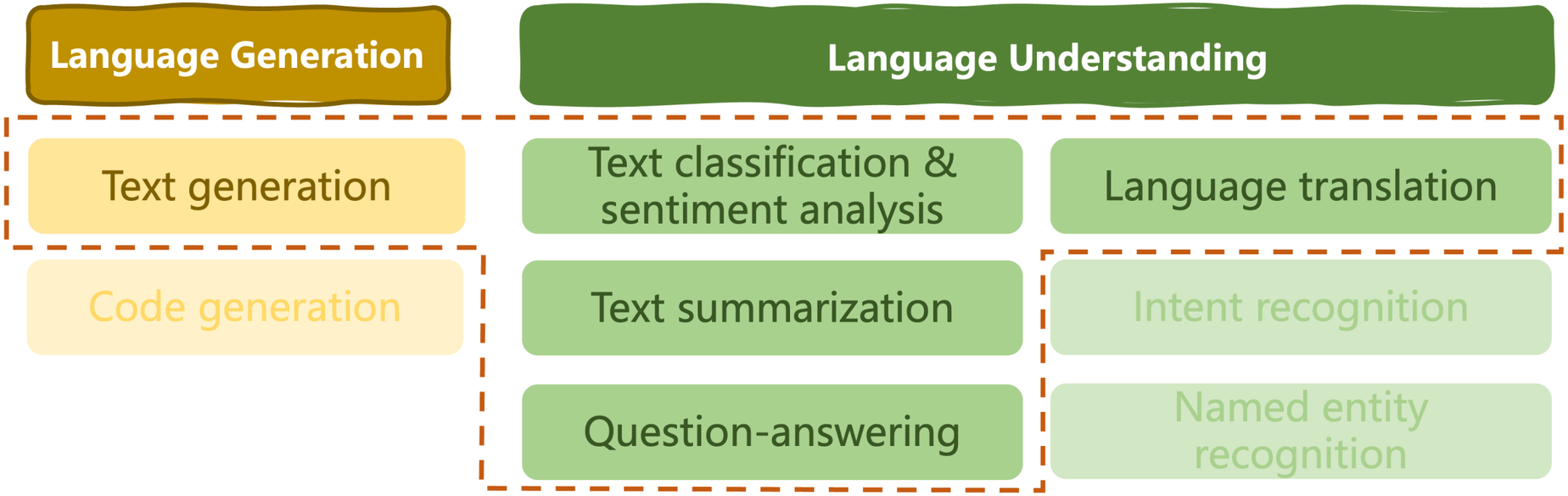
Language Generation

- Text generation
- Code generation

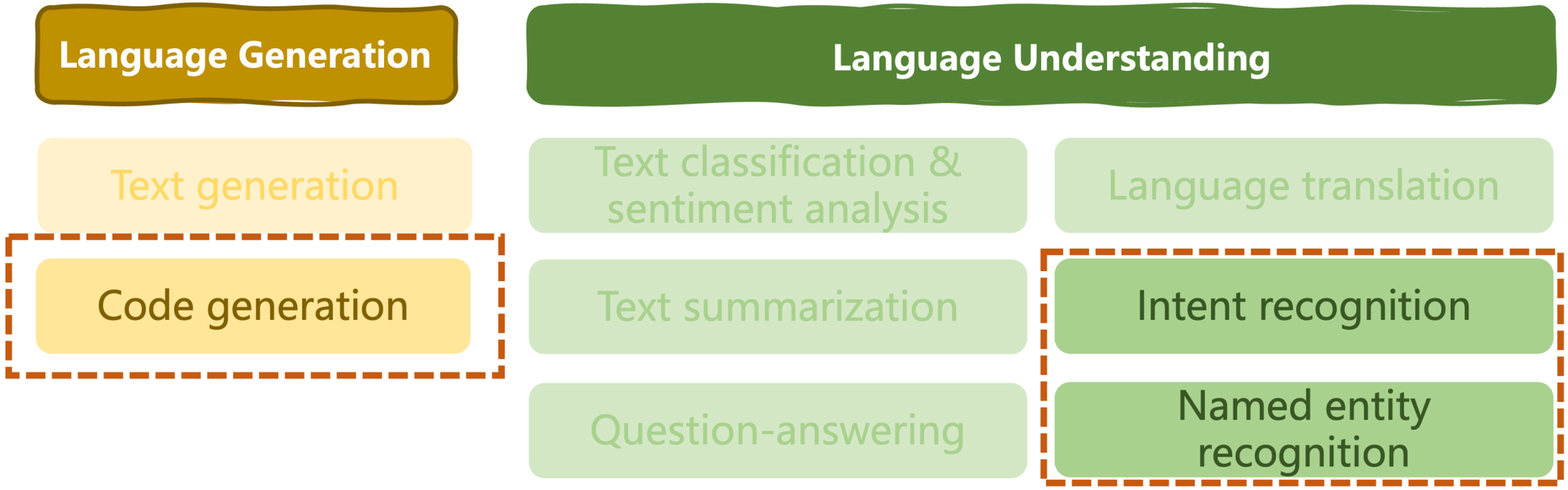
Language Understanding

- 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


```
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

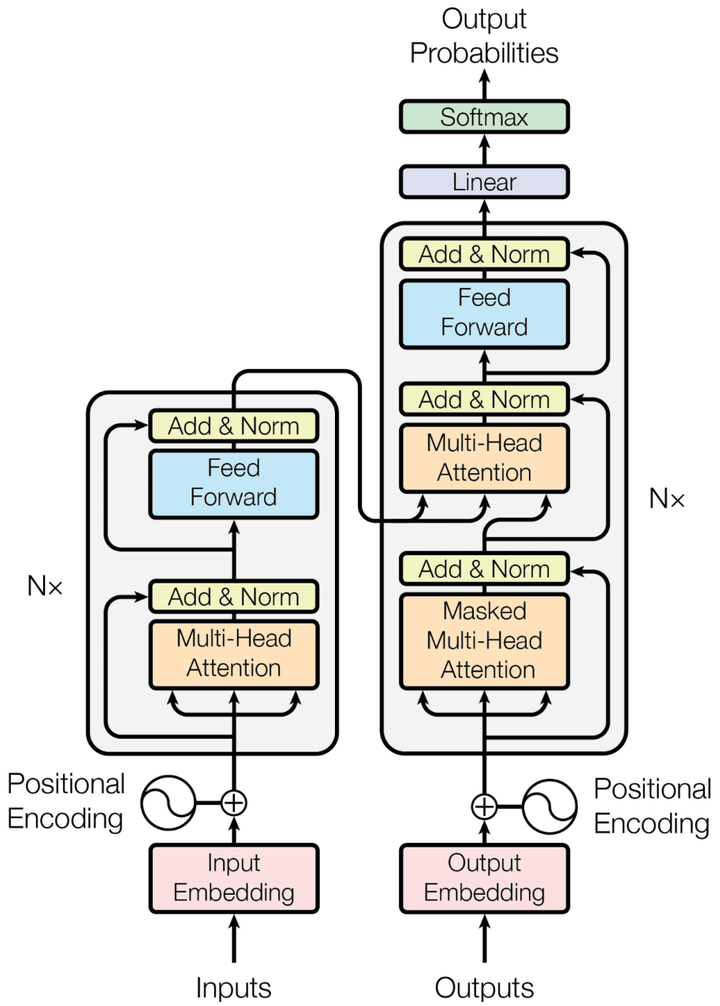
What is a transformer?

- **Deep learning** architecture for **text processing, understanding**, and **generation**

Characteristics:

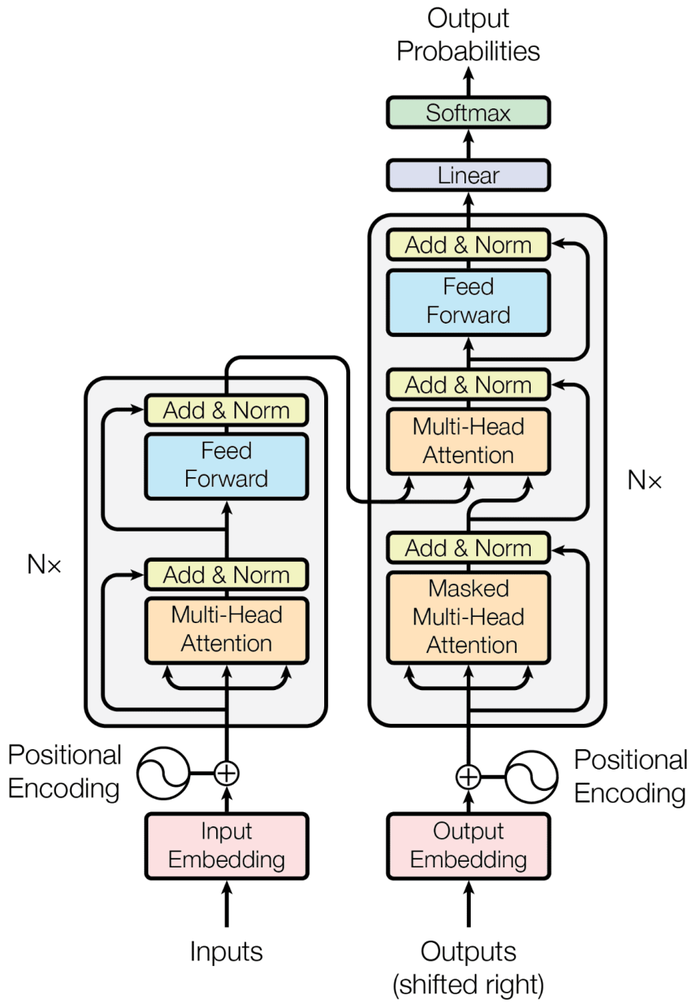
- No recurrent (RNN) architecture
- Capture **long-range dependencies** in text
- Tokens are handled simultaneously
- **Attention mechanisms + positional encoding**

The original transformer architecture



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

The original transformer architecture



- Two main stacks: **encoder** and **decoder**
- Each layer: attention mechanisms and feed-forward computations
- Capture complex semantic patterns and dependencies
- No **recurrence** nor **convolutions**
- Intended for various **language tasks**, e.g.:
  - Translation
  - Summarization
  - Question-answering

[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:

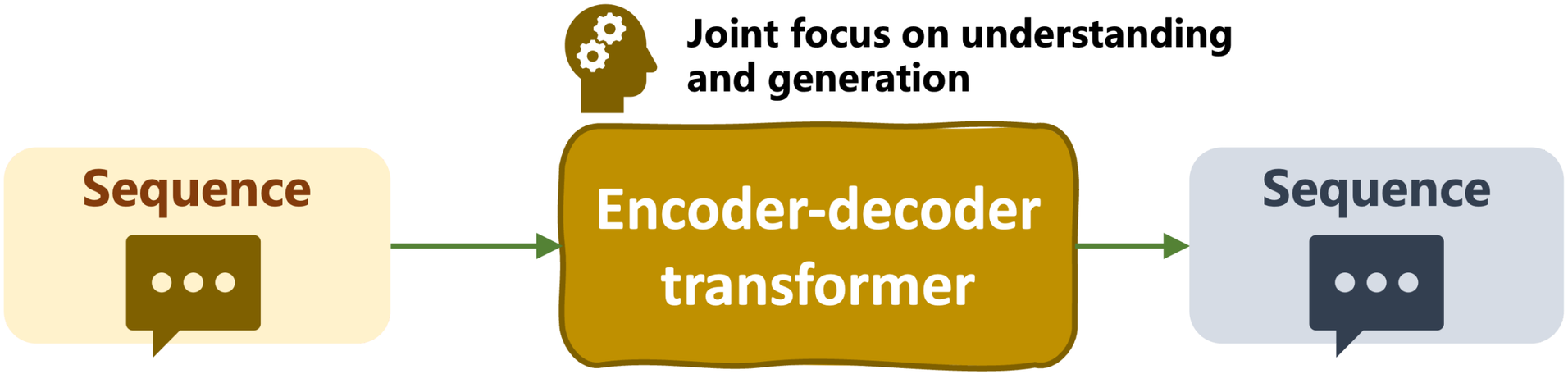
- `d_model` : model embedding dimension
- `n_heads` : number of attention heads
- `num_encoder_layers,num_decoder_layers` : number of encoder and decoder layers

PyTorch transformer initialization

- `torch.nn.Transformer` class

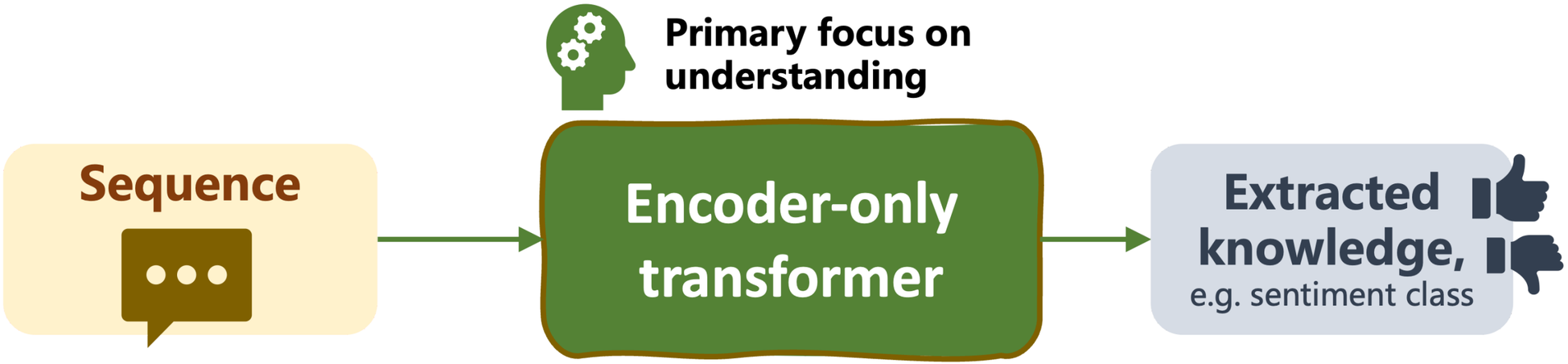
```
import torch
import torch.nn as nn

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



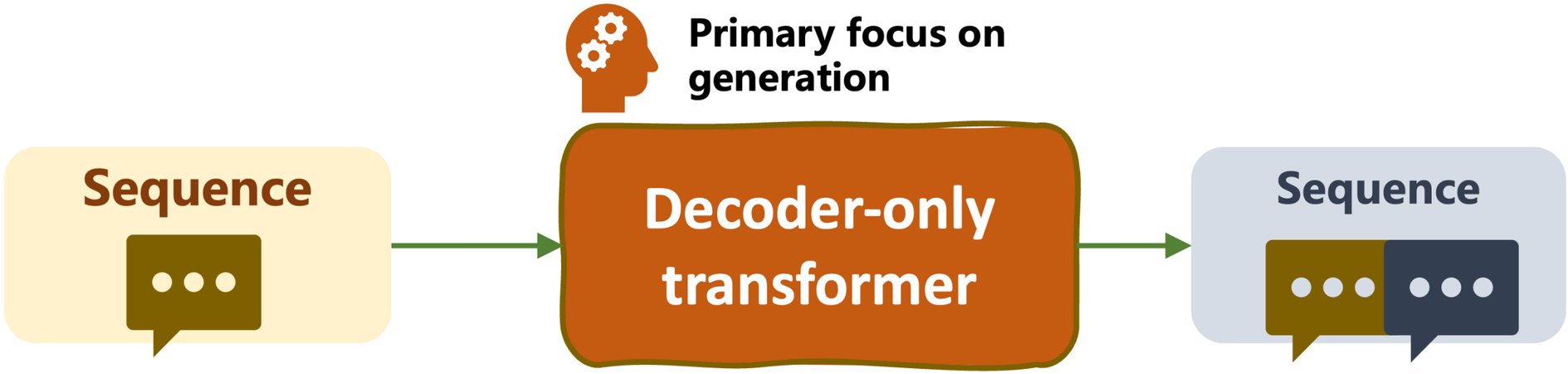
Type	Language tasks	Notable models
Encoder-decoder	Language translation, text summarization	T5, BART
-		
-		

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