Prompt Engineering

Generative AI

Module 1 – Lesson 3

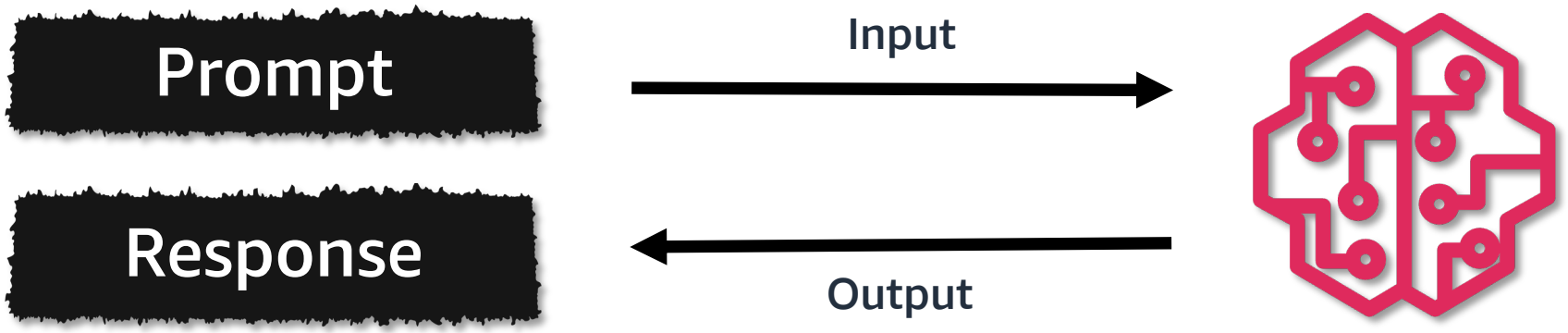
Today's activities

- Prompt engineering
- Inference parameters
- Best practices in prompt engineering
- In-context learning

Prompt Engineering

What are prompts?

- Inputs given to a model to get a response for a task
  - Typically, in the form of natural language query
- We can explain the task, set constraints, show examples or set the output format for the model's response
- Provides user intent for the model to generate the desired response



Components of a prompt

Let’s consider the following prompt and identify the various components

The following is a customer email received last week.

Summarize the main points of the email in a bulleted list.

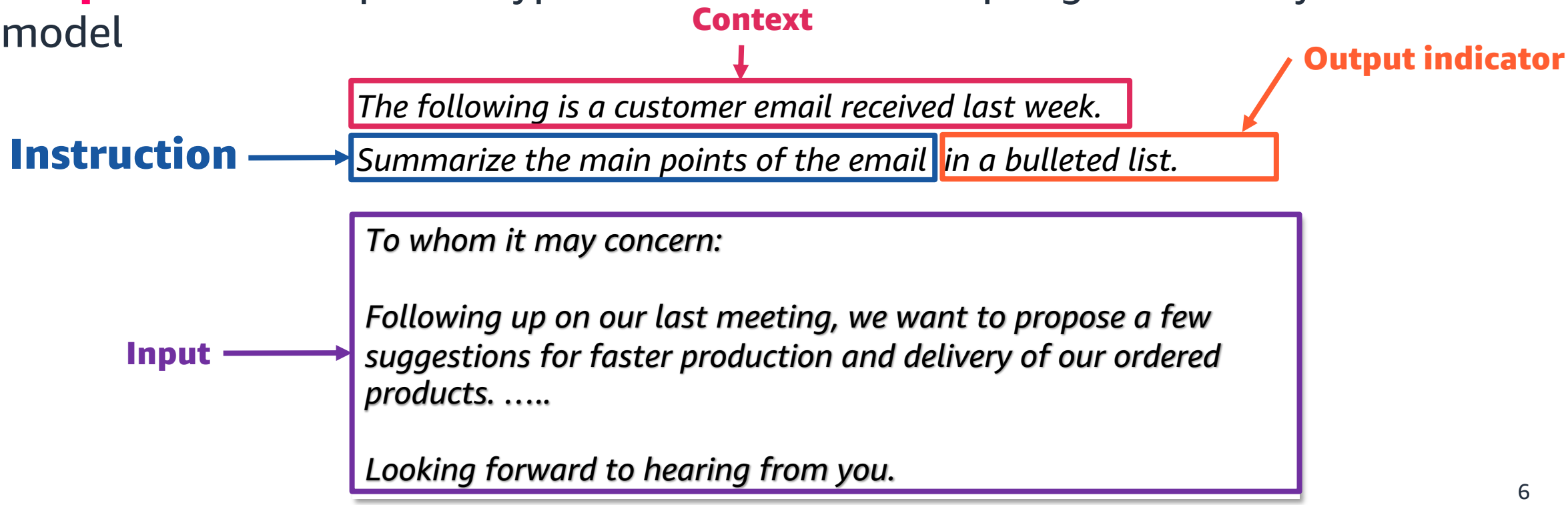
To whom it may concern:

Following up on our last meeting, we want to propose a few suggestions for faster production and delivery of our ordered products. …..

Looking forward to hearing from you.

Components of a prompt

- **Input:** Input data for the model
- **Instruction:** Task or instruction for the model (e.g. classify, summarize, …)
- **Context:** Additional relevant information to guide the model's response
- **Output Format:** Specific type or format of the output generated by the model



Prompt Engineering

Systematic design and optimization of prompts to guide the response of LLMs, ensuring accuracy, relevance, and coherence in the generated outputs.

- **Iterative process**. It can take multiple iterations to find optimal prompts
- Prompt quality and structure can significantly **influence performance**
- Well-constructed prompts can **counteract hallucinations**
- **Dynamic field, rapidly evolving**. Prompt engineering encloses different approaches, from best practices to emerging research techniques

Inference parameters

Inference Parameters

- **Inference parameters** help control and customize the model's response
  - Does not affect model architecture or weights
  - Control output generation during inference

- Affect properties such as:
  - Creativity and diversity in responses
  - Confidence of the response generation
  - Response length
  - Determine end of response generation

Inference parameters

- **Temperature**: controls the randomness
  - Use T=0 to make the output *deterministic*
  - Use higher temperatures to generate more *diverse* text and control creativity
- **Top p**: selects next word from those with probabilities summing up to said value
- **Top k**: picks up next token from the top “k”, sorted by probability
- **Maximum number of tokens**: controls the length of the generated response
  - May result in incomplete responses if value too low
- **Stop sequences**: stops generating text when it encounters these sequences

Best Practices in Prompt Engineering

Good prompting practices

- Write **clear** and **specific** instructions (unambiguous and specific)
- Highlight or specify the part of the prompt **that the model should focus** on
- Add relevant **details** or **restrictions** to your prompt
- Separate the instruction, content, question, and output directions
- Prefer using positive instructions
- Finding the optimum prompt is usually an **iterative** process which may take a few attempts

Prompts for instruction-tuned models

- LLMs are pre-trained on vast amounts of data
  - They have learned patterns, grammar, facts, and some reasoning capabilities
- **Instruction-tuned LLMs** are fine-tuned to predict **responses to textual instructions**
  - They can generate content that aligns closely with user intent
  - They follow guidelines to arrive at desired outcomes

Create an Amazon Bedrock user manual. Highlight its key features and explain the API implementation.

Model-specific prompts

- Different large language models might need specific prompt formats
- Refer to the model cards or documentation for prompting guidelines
  - **Example**: Anthropic's Claude model has been trained on alternating "Human" / "Assistant" dialogue. This needs to be replicated in the prompt.
  - **Example**: Open Assistant uses specific tokens to mark parts of the prompt

```
¶
¶
Human: Why are flamingos pink?
¶
Assistant:
```

Claude prompt

```
<|prefix_begin|>You are a large
language model that wants to be
helpful.<|prefix_end|>
<|prompter|>Why are flamingos
pink?<|endoftext|><|assistant|>
```

Open Assistant prompt

More inference strategies

- Quantization
  - Load model weights in lower precision data type
  - Lower memory consumption and compute requirements
  - Faster inference
  - Minimal loss in performance if applied effectively
- Use batch predictions over iterative single prediction
  - Faster inference, especially when using GPU

Cost effective strategies when querying API (1/2)

- Be aware of the **cost structure** when querying LLMs through APIs
  - Length of the prompt (# tokens)
  - Length of the response (# tokens)
  - Cost per query
- **Rule of thumb**: 1 token corresponds to ~4 characters of text
  - 100 tokens corresponds to 75 words

Cost effective strategies when querying API (2/2)

- Control the length of the response
  - Set inference parameters such as ’max_new_tokens’
  - Adding instructions in the prompt
    - For example: ‘Be concise’ or ‘answer in less than 50 words’
- Shorten prompts or combine multiple prompts
- Test more cost effective LLMs
  - Smaller LLMs or cheaper APIs

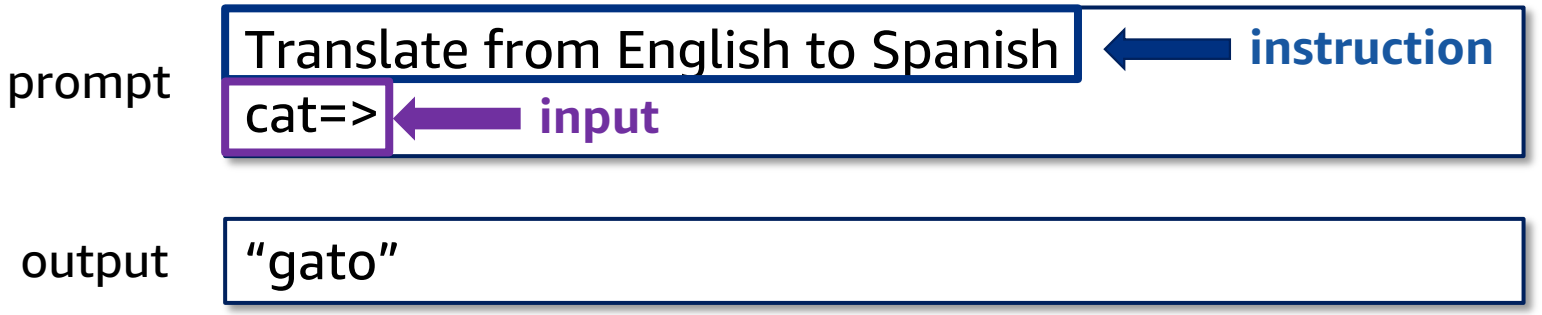
In-context Learning

In-context Learning

- In-context learning is a common adaptation method where the model is **not updated**, but provided with some instructions for different tasks
- For example, LLMs can be asked to do text classification, summarization and question/answering
- In addition to the instruction, the model can be given some correct examples of the task

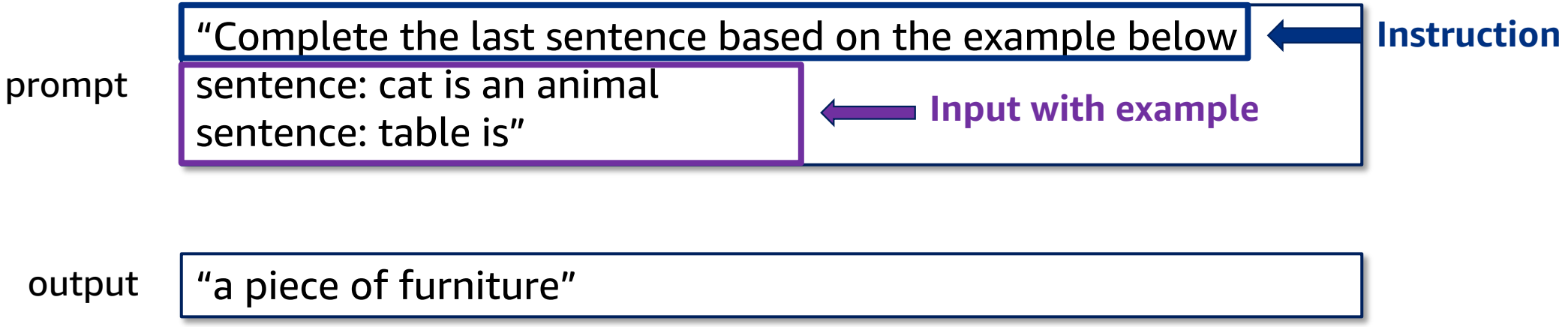
Zero-shot learning

- Generate a response solely based on an instruction
- Perform tasks that were not the core training objective of predicting next word
  - For example: Translation, summarization, arithmetic reasoning, etc.
- Leverage the generalized understanding of concepts from pre-training
  - **Emergent abilities**



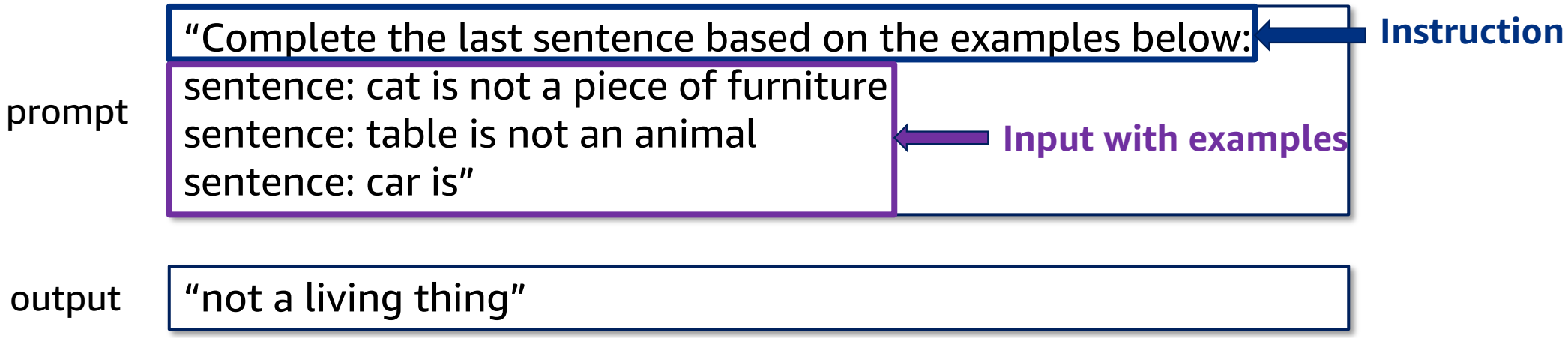
One-shot learning

- Cases where **one example** and the **instruction** are provided
  - Show the model how to perform the task



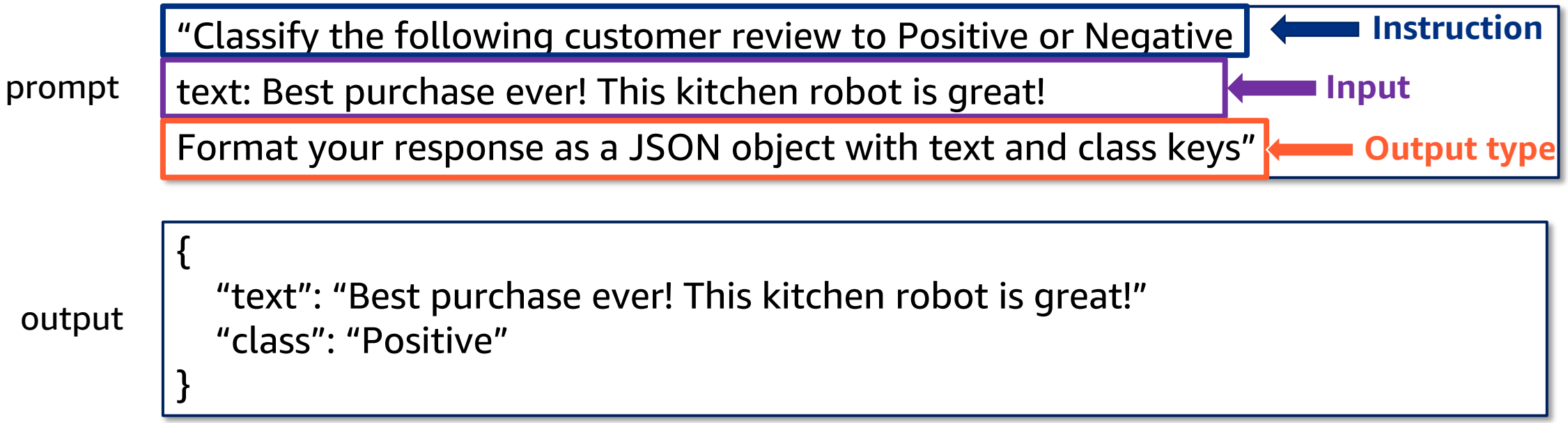
Few-shot learning

- Cases where **multiple examples** and the **instruction** are provided
- Identify patterns and apply it directly



Example: Sentiment analysis with zero-shot inference

- **Sentiment analysis**: Identify and extract **subjective information from text data**, such as opinions, emotions, attitudes and feelings



Example: Text summarization with zero-shot inference

- **Text summarization:** Generate a **summary of a larger text document or set of documents**, typically with the goal of capturing the most important information in a condensed form

prompt

"Summarize the following text in one sentence
text: It all started 5 years ago when EV sales started to go up quarter-by-quarter in all states in the US. It seemed like a major change in customer behavior, one change that would definitely benefit some other industries beyond the EV producers that……long text …… was the end of it."

output

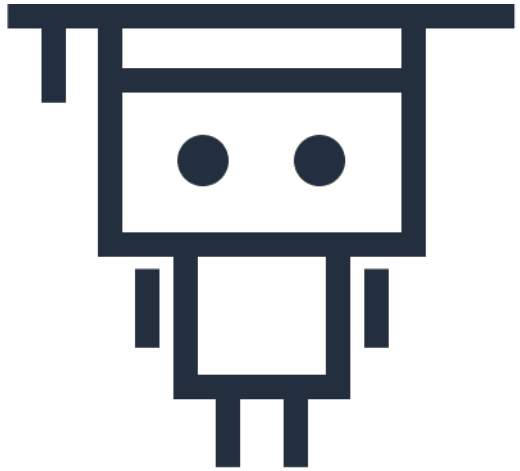
"Although recent increase in the EV sales had a positive impact in the number of charging stations built in the US, more support from the local agencies can accelerate the process."

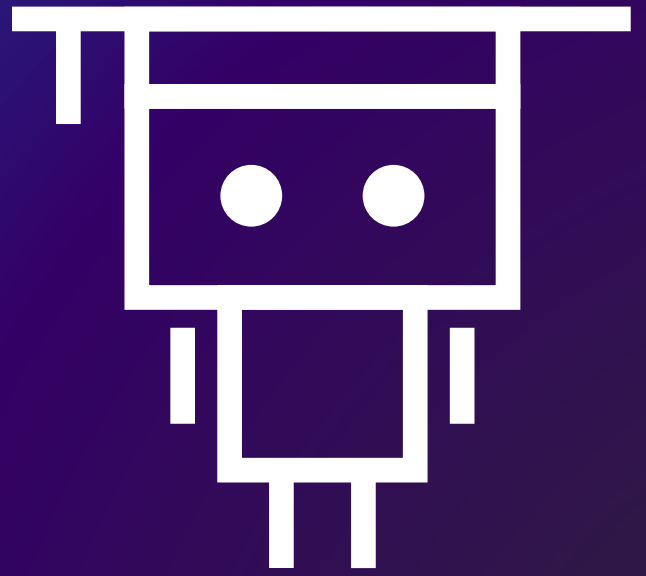
More zero-shot examples

- Personalized content generation
  - “Use 2-3 sentences to explain the in-context learning concept to a high school student.”
- Code generation
  - “Write Python code to read a CSV file”
- Information extraction
  - “What is the year mentioned in the following text? Text: ‘Amazon was the second largest employer in the US in 2022’”
- Question-answering tasks
  - “How many days are there in a year?”
  - “What is the capital of Italy?”

Next lesson

- This lesson covered prompt engineering and in-context learning
- In the next lesson, you will explore some advanced prompting techniques





Thank you!