

# Prompt Engineering



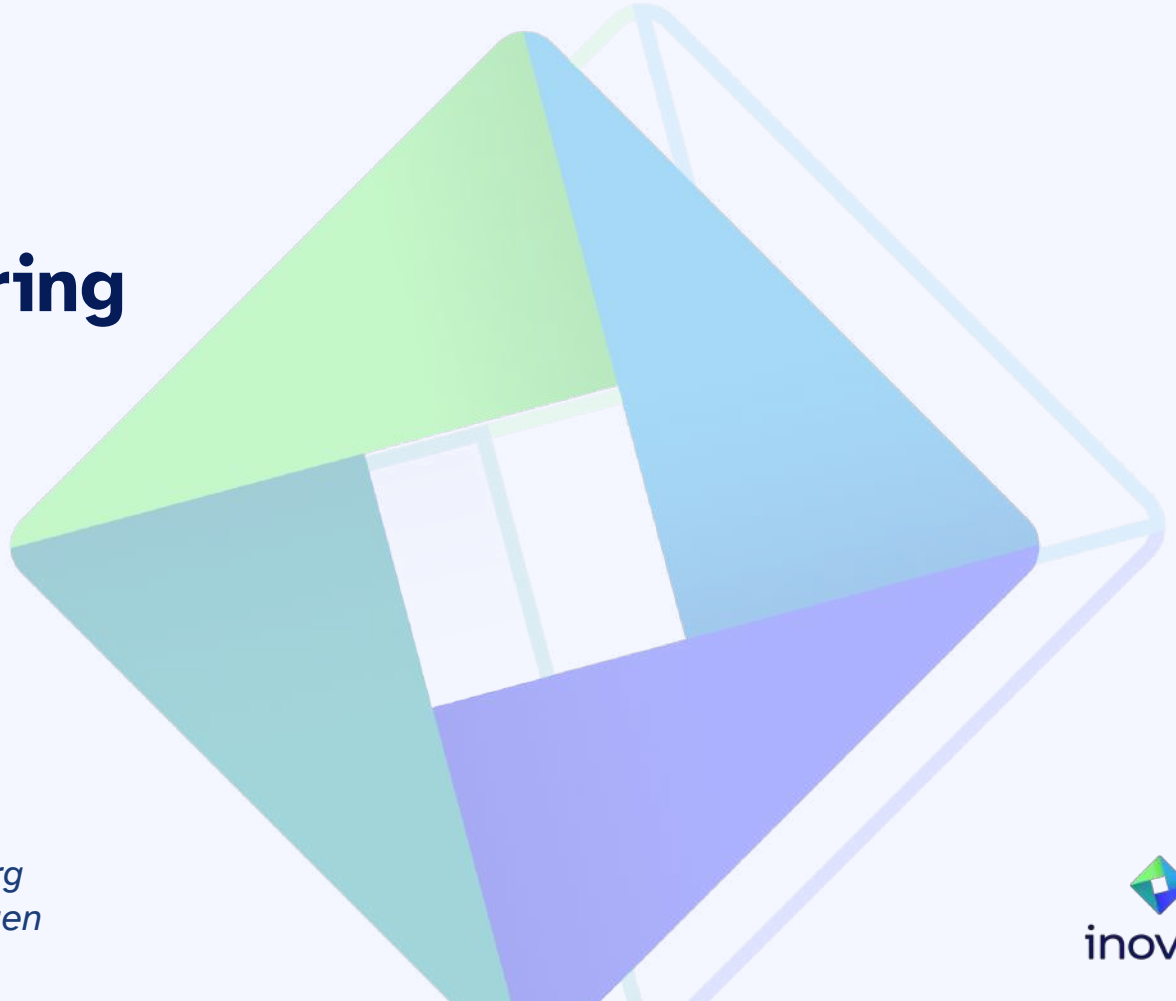
Prompt: A vintage 1980s high school yearbook photo of astonished child with open mouth

# Prompt Engineering

From Zero to GenAI Hero

## Team inovex

*Karlsruhe · Köln · München · Hamburg  
Berlin · Stuttgart · Pforzheim · Erlangen*



inovex

# Prompting = Programming with natural language



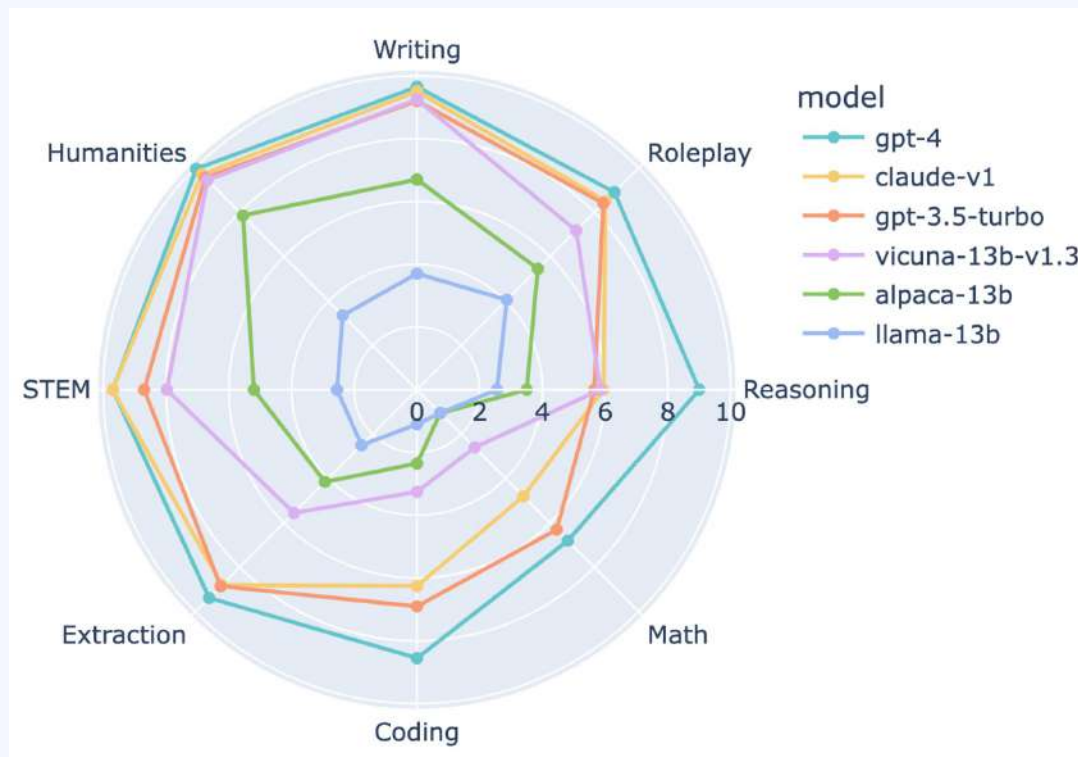
**Prompt:** A message you send to a GenAI

**Prompt Engineering:** The “science” to send the exact right message to get the output you want

# LLMs supercharged with Prompt Engineering

**“Let's think step by step”**

# LLMs are not good at math



# LLMs supercharged with Prompt Engineering

## (c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 X

## (d) Zero-shot-CoT (Ours)

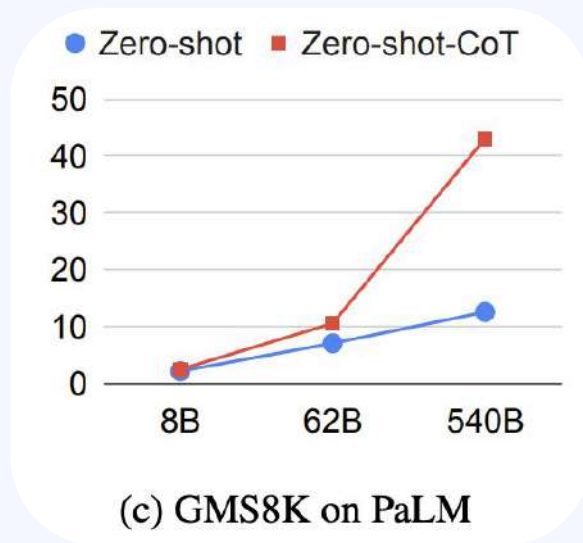
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) *There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓*

Paper: [Large Language Models are Zero-Shot Reasoners](#)

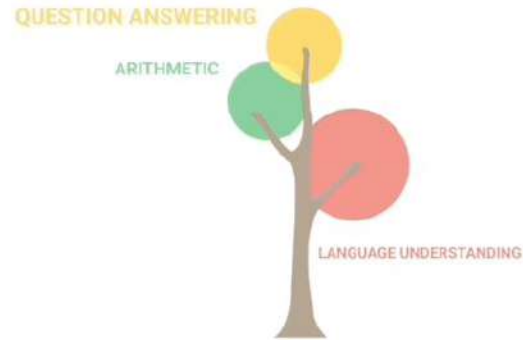
# LLMs supercharged with Prompt Engineering



Paper: [Large Language Models are Zero-Shot Reasoners](#)



# LLMs supercharged with Prompt Engineering



**8 billion parameters**

Source: <https://blog.research.google/2022/04/pathways-language-model-palm-scaling-to.html>

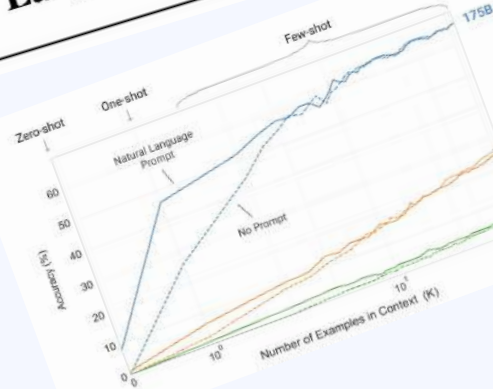
# LLMs supercharged with Prompt Engineering

Large Language Models are Zero-Shot Reasoners

Language Models are Few-Shot Learners

Reflexion: Language Agents with Verbal Reinforcement Learning

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models



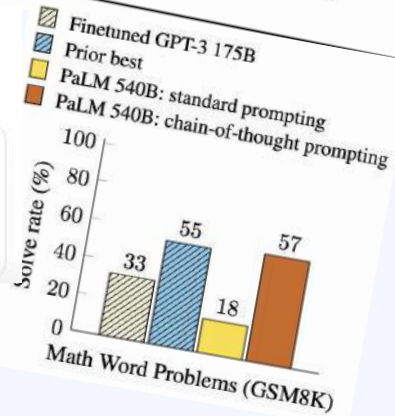
 Andrej Karpathy   
@karpathy

Good example of us not seeing max GPT-4 capability yet, imo. Prompt design, tool use, meta cognition strategies (eg idea of attempt, critique, retry, capabilities model, etc) are very likely to go a long way.

 Eric Jang @ericjang11 · Mar 26

Instead of finding the perfect prompt for an LLM (let's think step by step), you can ask LLMs to critique their outputs and immediately fix their own mistakes. Here's a fun example:

[Show this thread](#)



## Interest awakened



## Work routine so far

Goal: Creating a watercolor seagull image

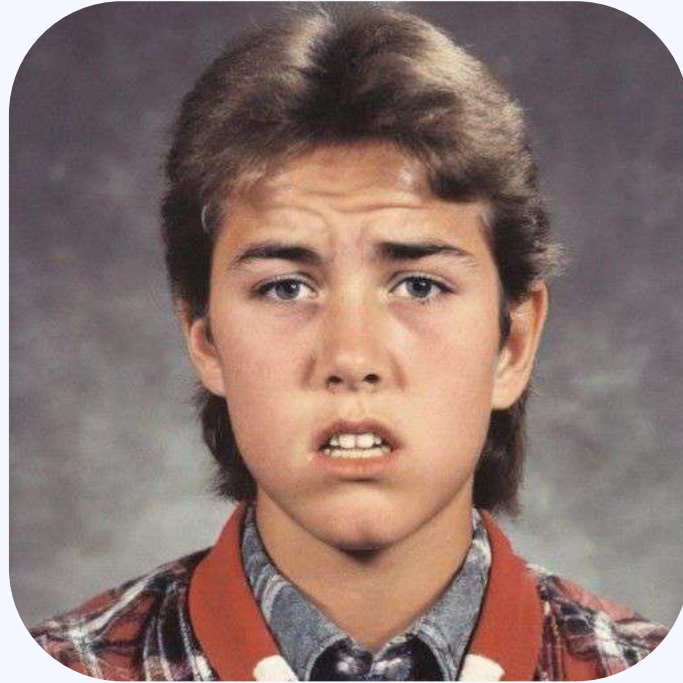


# Work routine so far

- 1 **Find** a suitable image
- 2 **Understand** GUI
- 3 **Remove** background
- 4 **Add** watercolor effect



## Work routine so far



## Work routine with GenAI

Prompt: Watercolor artwork of a flying seagull, white background, colorful seagull



## Work routine with GenAI



@sanderhoogeland9161 vor 1 Monat

I have had access to GPT-4 Vision for a few days now, and as a blind person, I must say that I really enjoy using it. Combined with the custom instructions I gave GPT-4, it explains images of scenes, areas, and items around me in a way that I really appreciate and is quite useful so far.



118

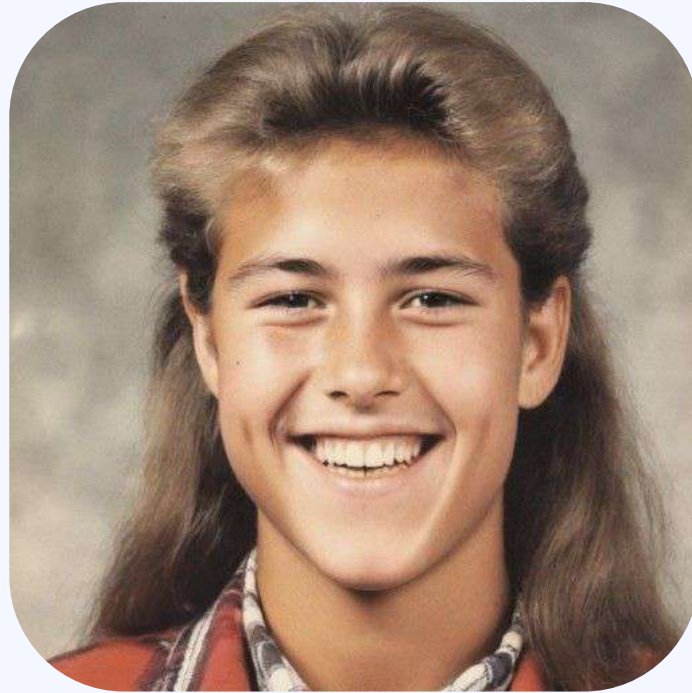


Antworten

**Accessibility** through natural language



## Work routine with GenAI

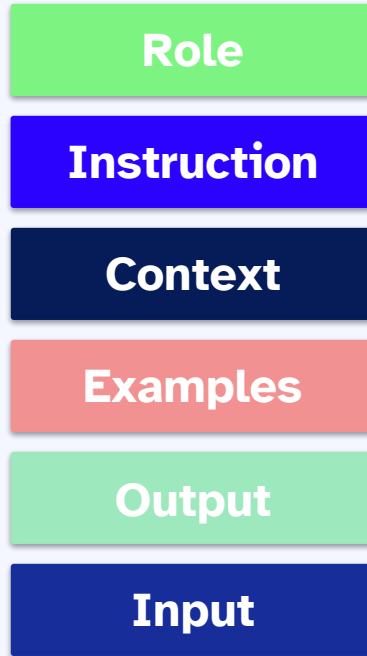


# **Prompt Engineering** as the key to **generative AI**

**Let's dive into it**



# How many elements does a Prompt contain?



# Prompt Element: Instruction

## Input

**Explain Prompt Engineering**

## Output

Prompt engineering refers to the process of designing and optimizing prompts in order to elicit desired responses from a language model like ChatGPT. It involves carefully crafting instructions or questions given to the model to guide its behavior and generate more accurate or desirable outputs. [...]

\*GPT-3.5

Techniques: Zero-Shot, Chain of Thought

## Prompt Element: Role

### Input

**You are a  
kindergartner.  
Explain Prompt  
Engineering.**

### Output

**Imagine that you are playing a game with your friends, and you want them to do something specific. Prompt engineering is like giving your friends really good instructions or clues to help them know what to do in the game. [...]**

\*GPT-3.5

# Prompt Element: Context

## Input

### **Explain Prompt Engineering with help of the context:**

Prompt Engineering, also known as In-Context Prompting, refers to methods for how to communicate with LLM to steer its behavior for desired outcomes without updating the model weights. It is an empirical science and the effect of prompt engineering methods can vary a lot among models, thus requiring heavy experimentation and heuristics.

## Output

**Prompt Engineering, or In-Context Prompting, is a technique used to influence the behavior of large language models (LLMs) without altering their underlying weights or parameters. [...]**

\*GPT-3.5

# Prompt Element: Examples

## Input

Find the right superordinate term of Prompt Engineering, by only considering "Data Management and Analytics", "Application Development" and "IT Engineering" as potential answers.

Techniques: Few-Shot/One-Shot Learning, Chain of Thought

## Output

The right superordinate term for Prompt Engineering would be "IT Engineering".

\*GPT-3.5

## Output

"IT Engineering"

\*GPT-4



# Technique: One Shot Learning

## Input

Find the right superordinate term, by only considering "Data Management and Analytics", "Application Development" and "IT Engineering" as potential answers.

**Human: Superordinate term of Natural Language Processing**

**AI: Data Management and Analytics**

**Human: Superordinate term of Prompt Engineering**

## Output

**Application Development**

\*GPT-3.5/  
GPT-4

# Technique: Few Shot Learning

## Input

Find the right superordinate term, by only considering "Data Management and Analytics", "Application Development" and "IT Engineering" as potential answers.

**Human: Superordinate term of Frontend**

**AI: Application Development**

**Human: Superordinate term of Generative AI**

**AI: Data Management and Analytics**

**Human: Superordinate term of Cloud**

**AI: IT Engineering**

**Human: Superordinate term of Natural Language Processing**

**AI: Data Management and Analytics**

**Human: Superordinate term of Mobile Apps**

**AI: Application Development**

**Human: Superordinate term of Prompt Engineering**

## Output

**Application Development**

\*GPT-3.5

## Output

**Data Management and Analytics**

\*GPT-4

# Technique: One-Shot Chain of Thought

## Standard Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The answer is 27. ❌

## Chain-of-Thought Prompting

### Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

# Prompt Element: Input/ Question

## Input

Find the right superordinate term, by only considering "Data Management and Analytics", "Application Development" and "IT Engineering" as potential answers.

Human: Superordinate term of Frontend

AI: Application Development

Human: Superordinate term of Generative AI

AI: Data Management and Analytics

Human: Superordinate term of Cloud

AI: IT Engineering

Human: Superordinate term of Natural Language Processing

AI: Data Management and Analytics

Human: Superordinate term of Mobile Apps

AI: Application Development

Human: **Superordinate term of Prompt Engineering**

## Output

**Application Development**

\*GPT-3.5

# Prompt Element: Output

## Input

Explain Prompt Engineering and define the subordinate term.  
**Format the output as JSON e.g.**

```
{  
  "explanation": "",  
  "subordinate_term": ""  
}
```

## Output

```
{  
  "explanation": "Prompt engineering is the process of [...]",  
  "subordinate_term": "Prompt"  
}
```

\*GPT-3.5

## Finally full-grown

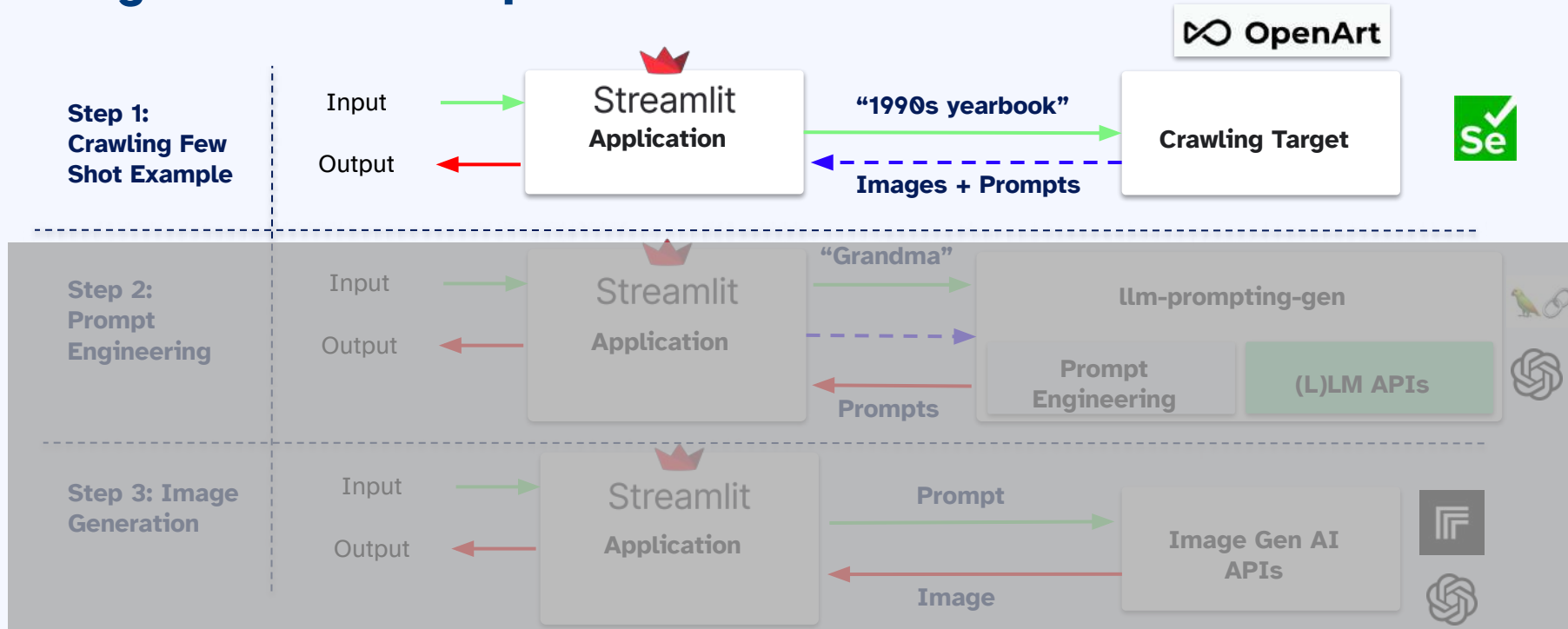


**How to apply our acquired knowledge to a specific use case?**

**Idea: Using prompt engineering techniques  
to create text-to-image prompts**

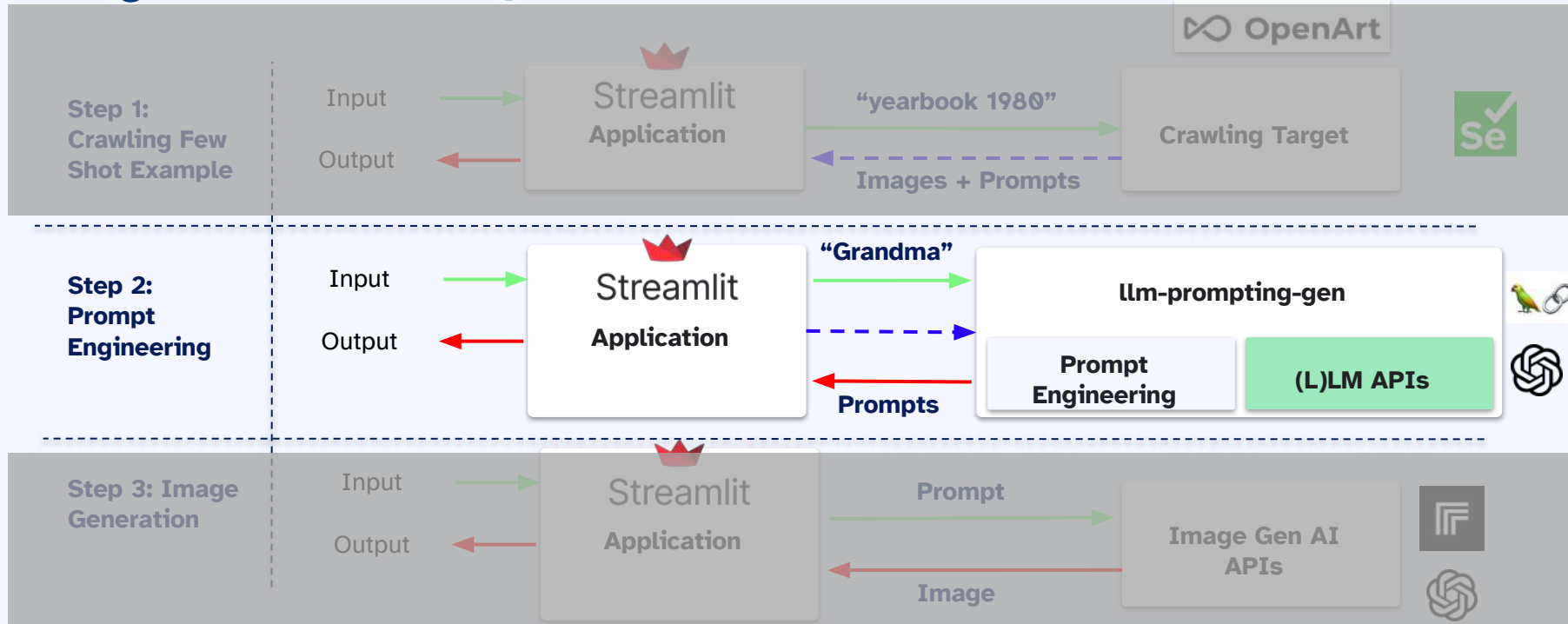


# Image Gen AI Prompt Generator



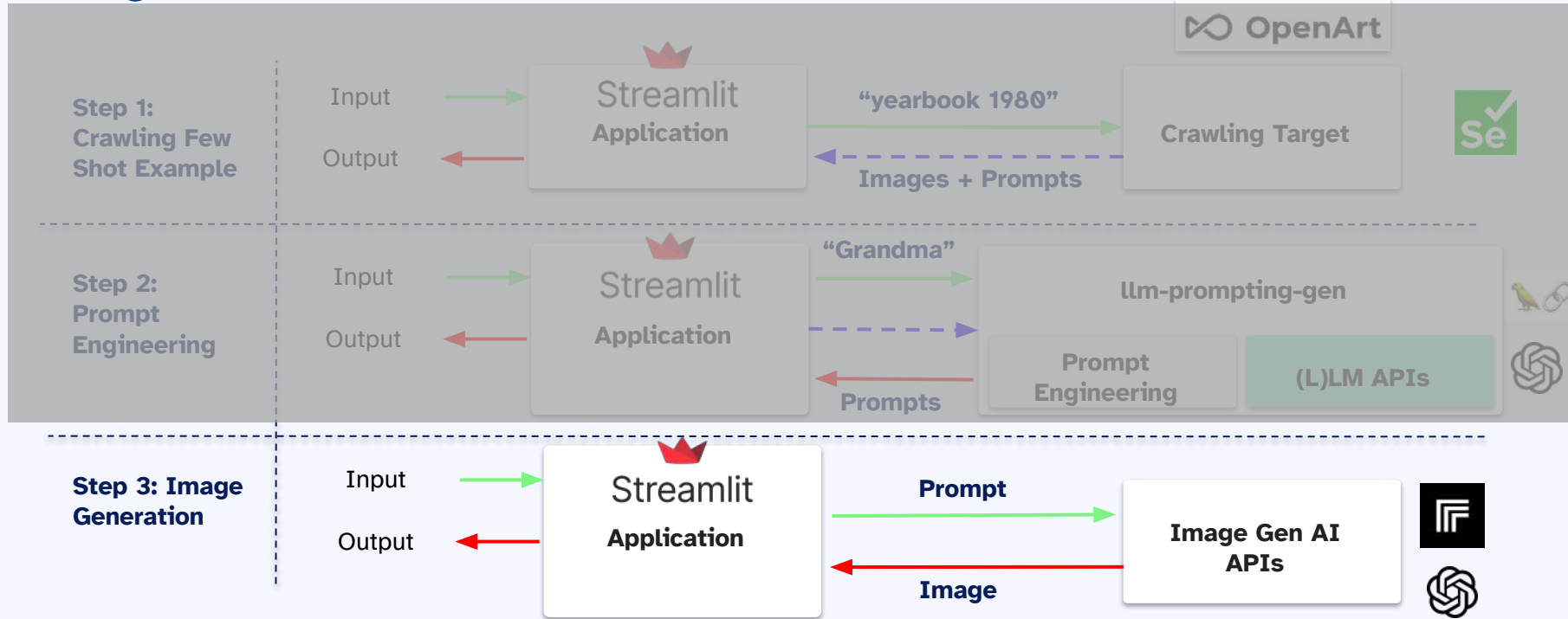
- Flow of crawling data
- Flow of user input
- Outputs returned to users

# Image Gen AI Prompt Generator



- Flow of crawling data
- Flow of user input
- Outputs returned to users

# Image Gen AI Prompt Generator



- - -> Flow of crawling data
- Flow of user input
- Outputs returned to users

# Image Gen AI Prompt Generator



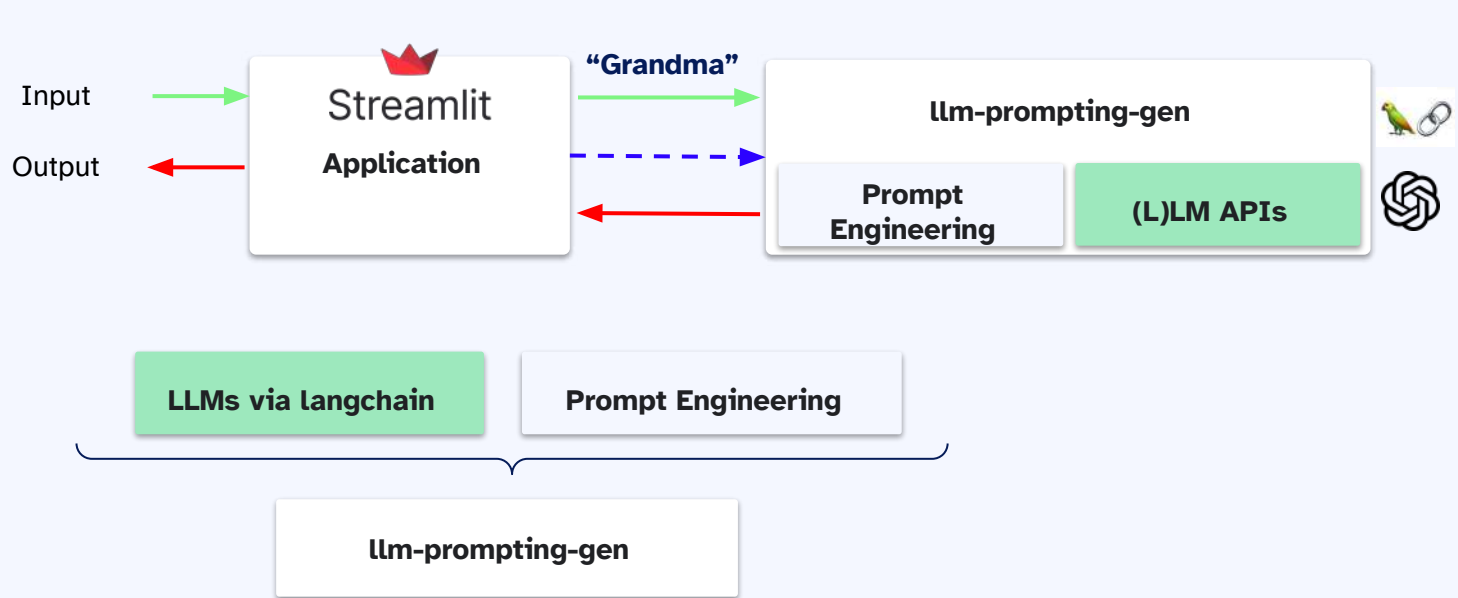
\*SDXL



\*Midjourney

# LLM Prompt Engineering Generator

Step 2:  
Prompt  
Engineering



Github: <https://github.com/FloTeu/llm-prompt-engineering-generator>

# Simple example

LLMs via langchain

Prompt Engineering

llm-prompting-gen

Template JSON stored in: "templates/kindergartner.json"

```
{
  "role": "You are a kindergartner.",
  "instruction": "Answer as you would speak to your kindergarten children.",
  "input": "{text}"
}
```

```
from llm_prompting_gen.generators import PromptEngineeringGenerator
from langchain.chat_models import ChatOpenAI

# Simply load a JSON file following the format of llm_prompting_gen.models.prompt_engineering.PromptElement
# Make sure env variable OPENAI_API_KEY is set
llm = ChatOpenAI(temperature=0.0)
keyword_extractor = PromptEngineeringGenerator.from_json("templates/kindergartner.json", llm=llm)
llm_output = keyword_extractor.generate(text="Explain Prompt Engineering")
```

Github: <https://github.com/FloTeu/llm-prompt-engineering-generator>

# LIVE DEMO



<https://image-gen-ai-app.streamlit.app/>



# Sources

## Prompt Engineering Websites

[learnprompting.org](https://learnprompting.org)

<https://www.promptingguide.ai/>

[Lil'Log](#)

## Papers

CoT: <https://arxiv.org/pdf/2201.11903.pdf>

Few Shot Learners: <https://arxiv.org/pdf/2005.14165.pdf>

Reflexion: <https://arxiv.org/pdf/2303.11366.pdf>

Zero Shot Reasoners: <https://arxiv.org/pdf/2205.11916.pdf>

## Key takeaways

1. 6 prompt elements are all you need
2. Write clear, clean, and detailed instructions
3. Give the model time to think

# What's Next?

27.02.2024:

What the  
Web?  
HTMX -  
Return to  
Monkey?



Thanks!



**Florian Teutsch**

Machine Learning Engineer



Florian Teutsch



@inovexgmbh



@inovexlife